Chapter 2

Fun Projects Using Sage

Here are some extended examples of using Sage to explore a non-trivial problem in an area outside of computer algebra. We present one example from each of several disciplines. The idea is that these projects would each take a typical student about one long weekend to complete.

Some of these projects can be explored using information taught in Chapter 1 and do not require any additional familiarity with Sage. Some projects do require a bit more, but that is clearly marked in the first few paragraphs of the project description.

Microeconomics: Modeling the effect of the selling price on the sales of ice cream, in terms of a demand function, a revenue function, a cost function, and a profit function.

Biology: Modeling clogged arteries and Poiseuille's Law, and the catastrophic effects of even small percentages of blockage on blood flow. No knowledge of biology is assumed.

Industrial Optimization: Solving a linear system of inequalities to optimally route Taconite from 7 mines to 4 ports for shipping, while minimizing shipping costs and respecting capacities.

Chemistry: A method for balancing rather complicated chemical reactions by using the RREF of matrices.

Physics: Solving ballistic projectile motion problems in Sage, including counter-battery fire.

Cryptology: Using Pollard's $p - 1$ Method of factoring integers to try to break the RSA cryptosystem.

Pure Mathematics: Using differential calculus to analyze a quintic polynomial's features for finding the optimal graphing window.

Electrical Engineering: A mini-project on generating electric field vector plots for the field generated by four charged particles in the coordinate plane.

Also, I am writing a book for a course commonly called *Finite Mathematics* in the USA and *Quantitative Methods* in the UK. This is a course for people majoring in business, accounting, finance, economics, marketing,

management, and so forth. A classic topic in that course is the Leontief Input-Output Analysis method of modeling how various sectors of the economy interact. This would be a topic in macroeconomics, to complement the microeconomics project given here. While it does require a bit of matrix algebra, Leontief Analysis is very well suited to Sage and relatively easy to pick up. You can find a version of that project on my webpage:

`www.gregorybard.com`

2.1. Microeconomics: Computing a Selling Price

I was once told that the goals of the first microeconomics course should be twofold: first, to teach the concept of "supply and demand" and second, to teach that the three actions of "maximizing profit," versus "maximizing revenue," or "minimizing costs," are three extremely different goals. This is an oversimplification, of course, and was probably meant as a joke. Nonetheless, it highlights the paramount importance of these two topics.

This project will look closely at the issue of "maximizing profit," versus "maximizing revenue," or "minimizing costs." It is an extended study of the microeconomics of the sale of ice cream. We imagine a vendor with an ice cream cart at a busy intersection in Central Park in New York City.

The ice cream vendor observes that when he charges \$3 per ice cream cone, he only sells 120 cones per day. On the other hand, when he charges \$2 per ice cream cone, he sells 200 cones per day. In our example, because he has long since paid off his start-up expenses, the costs of doing business are just 50 cents per cone, plus a \$100 per day fee for his permit to sell food in Central Park. As you might imagine, if he makes his price too high, he will sell too few cones and won't make much profit; if he makes his price too low, he will sell more cones, but he still won't make much profit.

The ice cream vendor is going to assume that the demand varies linearly with price. This usually is not quite true, but it is often a reasonable approximation. There are several different ways to realize that the demand n and the price p will have the relationship

$$n = 360 - 80p,$$

or equivalently

$$p = 4.50 - \frac{n}{80}.$$

Let's pause a moment and see if this makes sense. First, plugging in $n = 200$ and $n = 120$, we do get the prices of \$2 and \$3 that we expect. If we set $p = 2.50$, then we get 160, as we would anticipate—that's the midpoint.

The easy way to find those equations is to compute the slope connecting the points $(200, 2)$ and $(120, 3)$ and then find the equation of the line between those two points. Of course, the form of the equation depends on whether or not you assign p or n the role of x. I have seen both conventions used in textbooks—there is no "industry standard."

Now observe that, at a price of \$4.50, he will sell no ice cream at all. This should be believable—if people are usually expecting to pay around \$2, then you cannot charge them more than double what they are accustomed to pay. The price where demand is zero is called "the maximum feasible price." Meanwhile, if the price of a cone is \$0, then we sell 360 cones per day—that's called the saturation point and it is the most that one could ever hope to sell, namely the demand at the price of \$0. The model breaks down for prices over \$4.50 because it predicts a sale of a negative number of cones, which does not make sense. Likewise, if you set a demand above 360 cones per day (the "saturation point"), then a negative price is called for, which also does not make sense. So the model is valid for $0 \leq p \leq 4.50$ dollars, or equivalently $0 \leq n \leq 360$ cones.

We'd now like to make some revenue, cost, and profit functions in terms of the number of cones sold, n. Clearly if one sells n cones at price p, then the revenue is np. Therefore, we have

$$r(n) = np = n\left(4.50 - \frac{n}{80}\right) = 4.50n - \frac{n^2}{80}$$

as the revenue function. Meanwhile, the cost function is 50 cents per cone plus \$100, for a total of

$$c(n) = 0.50n + 100,$$

and that gives us a profit function of

$$p(n) = \frac{-n^2}{80} + 4n - 100$$

which is a parabola. It makes sense that we should get some sort of curve because if you make the price very low, or very high, then the profit is zero or worse—you have a loss. Somewhere in the middle is the sweet spot, where profit is made.

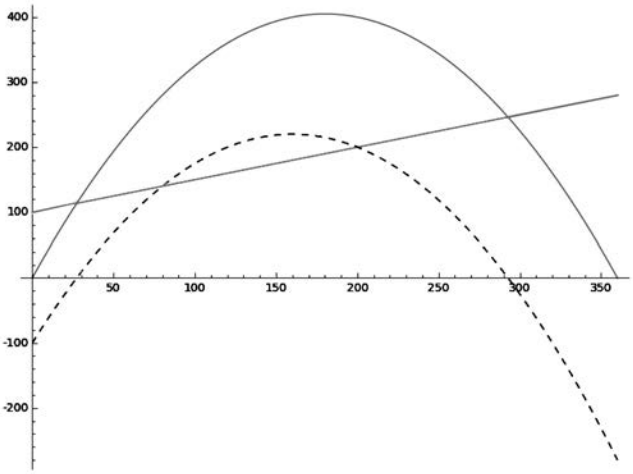


FIGURE 2.1. The plot for the microeconomics project.

Let's make a table and see that these functions make sense and match our prior data:

Price	Cones Sold	Revenue	Costs	Profit
\$0.00	360	\$0.00	\$280.00	−\$280.00
\$0.50	320	\$160.00	\$260.00	−\$100.00
\$1.00	280	\$280.00	\$240.00	\$40.00
\$1.50	240	\$360.00	\$220.00	\$140.00
\$2.00	200	\$400.00	\$200.00	\$200.00
\$2.50	160	\$400.00	\$180.00	\$220.00
\$3.00	120	\$360.00	\$160.00	\$200.00
\$3.50	80	\$280.00	\$140.00	\$140.00
\$4.00	40	\$160.00	\$120.00	\$40.00
\$4.50	0	\$0.00	\$100.00	−\$100.00

I recommend that you pick 2 or 3 values and manually verify those 2 or 3 rows of the table before continuing onward.

Okay, did you really verify 2 or 3 rows of the table? It is really useful to do so before reading further.

In any case, the graph of these three functions is given in Figure 2.1. As you can see, the profit function is the dashed parabola, the cost function is the straight line, and the revenue function is the solid parabola. The code to make Figure 2.1 is given on page 68. There are lots of facts that can be gleaned from this:

- When the cost function and the revenue function cross, that's the break-even point. In other words, when the cost and revenue are

equal, you break even. Here we have two break-even points, one just under 30 and one just under 300. To find them, we type

```
find_root(  (-x^2)/80 + 4.5*x == 0.5*x+100, 250, 350)
```

and also

```
find_root(  (-x^2)/80 + 4.5*x == 0.5*x+100, 10, 50)
```

to find out that at roughly 292.664 cones and 27.3350 cones, we break even. Between these values is a profit, and outside these values, there is a loss. That seems to match the data in the table.

- The other definition of the break-even point is when profit is zero. So we could have tried to find the roots of the profit function, as an alternative means of finding the break-even points. We would do this by typing

```
find_root(  (-x^2)/80 + 4*x - 100, 10, 50)
```

and also

```
find_root(  (-x^2)/80 + 4*x - 100, 250, 350)
```

to find those same values (292.664 and 27.3350), and you'll see that the dashed curve crosses the x-axis there.

- The cost function is never zero. That's because the ice cream vendor has to pay the \$100 per day fee. As you can see, the line never crosses the x-axis for any positive n.
- The revenue function is zero at two points. First, it occurs when no ice cream is sold—that makes sense. Surely we can agree that if no ice cream is sold, then there is no revenue. We found earlier that is at \$4.50 per cone and at 0 cones per day. Second, revenue is \$0 when the ice cream is sold for 0 cents per cone. Clearly, if you don't charge anything, then there is no revenue. That occurs at the saturation point of 360 cones per day and at \$0 per cone. To find these in Sage, we should type

```
find_root( 4.50*x-(x^2)/80, 0, 50)
```

and also

```
find_root( 4.50*x-(x^2)/80, 300, 400)
```

to get the values 0.0 and 360.0.

- Now we've found when profit is zero and when revenue is zero and determined that cost is never zero. The next step is to try to minimize or maximize these functions. If you've had calculus, then you know how to minimize or maximize a polynomial. If not, then you might recall the fact that a parabola $y = ax^2 + bx + c$ has its optimum at the point $x = -b/2a$.
- For the revenue function, we see that maximum revenue occurs at 180 cones, a cost of \$2.25 per cone, and a revenue of \$405. The costs would be \$190, for a profit of \$215.

- For the profit function, we see that maximum profit occurs at 160 cones, a cost of \$2.50 per cone, and a revenue of \$400. The costs would be \$180, for a profit of \$220.
- The point of minimum costs would be 0 cones of ice cream, for a cost of \$100. However, the revenue is \$0 and the profit is negative, a loss of \$100. Definitely, this is not a good business plan (even if the costs are lower in this bullet than in the previous two bullets).
- Did you notice how the point of maximum profit had lower revenue (\$400 vs \$405) but higher profit (\$220 vs \$215) than the point of maximum revenue? Did you notice how the point of maximum revenue had lower profit (\$215 vs \$220) but higher revenue (\$405 vs \$400) than the point of maximum profit?

In conclusion, we see that "maximizing revenue," "minimizing costs," and "maximizing profit," are three different objectives. Furthermore, we should suggest that the ice cream vendor sell his ice cream at \$2.50 per cone—because maximizing profit is the only intelligent thing to do.

By the way, the code to produce the image with the graphs of the three functions is as follows:

```
r(x) = 4.50*x - x^2/80
c(x) = 0.5*x + 100
p(x) = r(x) - c(x)

P1 = plot( r(x), 0, 360, color='green' )
P2 = plot( c(x), 0, 360, color='red' )
P3 = plot( p(x), 0, 360, color='black', linestyle='--' )
P = P1 + P2 + P3

P.show()
```

Note, profit is drawn in black, cost is drawn in red, and revenue is drawn in green, which together comprise a common coloring standard for problems of this kind.

Challenge:

Your challenge is to analyze a new business situation. A particular book is pre-sold by a publisher in two test markets for \$49.95 and for \$39.95. It sells 80 copies and 140 copies, respectively. In the past, the general sales for the year are 100 times the sales of one of the test markets. (Just to be clear, that means if the price is set at \$49.95, the marketing experts predict 8,000 copies will be sold, and likewise 14,000 copies at \$39.95.)

The books require \$7 to print plus \$4,000 payable once for the proof-reading costs. The publisher gets 50% of the sales revenue. In case you are curious, usually around 20% of the sales revenue goes to the local book-store, 20% to the distributor, and 10% goes to the author's royalties. You

may assume that the sales price and the number of copies sold have a linear relationship.

The deliverables for this project are

(1) an equation relating the sales price and the number of copies sold,
(2) the revenue function,
(3) the cost function,
(4) the profit function,
(5) the image showing all three functions graphed simultaneously,
(6) the break-even point(s),
(7) the n and p to result in maximum revenue,
(8) the n and p to result in maximum profit.

2.2. Biology: Clogged Arteries and Poiseuille's Law

If an American dies of natural[1] causes, the probability that it is of heart disease is 35.30%, of various forms of cancer is 33.94%, and of any other cause is 30.75%. Considering that heart disease kills roughly a third of us, the realities of clogged arteries and cholesterol should concern us all.

Blood flowing through an artery is governed by the same equation as water flowing in a pipe or oil flowing through a hose. This applies to all such flows of fluids through a cylinder, except for turbulent flow (where the fluid is churning around and tumbling over itself) and compressible flow (such as air flowing in a pneumatic hose). The equation is called Poiseuille's Law, named for Jean-Léonard Marie Poiseuille (1797–1869), whose dissertation[2] was about applying this mathematics to the human aorta.

In this project, we're going to explore the relationships between some of the variables in Poiseuille's Law. We are going to display those interdependencies through tables. In order to do this project, you need to have read Section 5.1, which describes how to make tables in Sage.

The Background:

The formula itself is

$$\dot{V} = (\Delta P)\frac{\pi r^4}{8\mu L}$$

and, regrettably, we do not have time to derive it here. However, I strongly recommend the curious reader to refer to the article "Blood Vessel Branching: Beyond the Standard Calculus Problem," by Professor John Adam, published in *Mathematics Magazine*, Vol. 84, by the Mathematical Association of America, in 2011.

[1]Based on 2010 data from the Center for Disease Control and Prevention's website "FastStats." This data excludes (as unnatural causes) any homicides, suicides, and accidents.

[2]Poiseuille, J-L. M., *Recherches sur la force du coeur aortique*, D.Sc., *École Polytechnique*, Paris, France, 1828.

Let's take a moment to identify the meaning of each of those variables.

- The $\dot{V}$ is the volume flow rate, in units such as cubic centimeters per minute. In a major artery, a typical flow might be 100 cm^3/minute. Note, $\dot{V}$ is another way of writing dV/dt.
- The radius of the artery is r, in units such as centimeters. A major artery might be 0.3 cm or very roughly 1/8 of an inch in diameter.
- The length of the artery is L, again in units such as centimeters. A major artery might be 20 cm or 8 inches in length.

Note: The unit of pressure can vary. The standard unit in the American system of units is pounds per square inch because forces are measured in pounds. Likewise, in the metric system, because forces are measured in newtons, the pressure units then become newtons per square meter. Using newtons per square meter is kind of funny because arteries aren't several square meters in area. Another unit of pressure used is "atmospheres" where atmospheric pressure is 1 atmosphere. That's 14.6959 pounds per square inch or 101,325 N/m^2.

- Medicine has been around for a long time. For historical reasons, it is normal to measure blood pressure in a very archaic unit called "millimeters of mercury." When you go to the doctor and are told that your blood pressure is "120 over 80," that 120 means "120 millimeters of mercury," abbreviated 120 mm Hg. Just take it as a unit and consider 120 to be "normal." If you have 145 mm Hg for the first number (called systolic), you might be diagnosed as having "hypertension." A score of 180 mm Hg systolic can result in organ damage and is called a "hypertensive emergency."
- Viscosity is a physics variable that you might or might not have seen before and is denoted μ. It represents how "sticky" or "thick" a fluid might be. Since blood is the fluid in question and we won't consider other fluids, we can just take μ to be a constant. The usual unit is a "poise." A typical value for blood might be $\mu = 0.0035$ poise. In comparison, the oil in your car might have $\mu = 0.250$ poise (because it is sticky) and water has $\mu = 0.001$ millipoise (because it is not sticky). Honey is very sticky and has a viscosity of 20 poise. We can't use the poise as our unit because we're using an antique unit for pressure. For us,

$$\mu = 4.37535\ldots \times 10^{-8}$$

is just a constant in the formula. The units of viscosity turn out to be "sec mm Hg," which is extremely strange, but that is what will make the correct units work out in the formula.

Your Challenge:

For this project, you're going to create some tables that can be used to help a physician understand the numerical realities of that quartic relationship in Poiseuille's Law between arterial radius and blood flow rate.

The tables will allow some variables to change, while most variables are locked at "standard values" that are chosen to be very typical. The standard values for our variables in this project are $\mu = 4.375359 \times 10^{-8}$ for the viscosity, $L = 20$ cm for the artery length, $r = 0.3$ cm for the arterial radius, $\dot{V} = 100$ cubic centimeters per second for the blood flow rate, and finally ΔP is 0.0275 mm Hg.

- Your first table should keep our standard values for μ, L, and ΔP and compute $\dot{V}$ based on r, using Poiseuille's Law. The r should be computed based upon the percent blockage, from 0% up to 50%, in 2% increments. The first column of the table should be the percent blockage; the second column should be the blood flow rate or $\dot{V}$.
- Your second table should keep our standard values for μ, L, and $\dot{V}$. Again, the r should be computed based upon the percent blockage, from 0% up to 50%, in 2% increments. This time, you will compute the ΔP required to achieve the flow rate of 100 cc/minute given the blockage (the $\dot{V}$ given the ΔP), using Poiseuille's Law. The first column of the table should be the percent blockage; the second column should be the ΔP. The third column should represent ΔP as the "percent of normal." In other words, 3/2 the normal ΔP should be reported as 150%.
- The third table should keep our standard values for μ, L, and ΔP. The first column should be the "percent normal flow" from 100% down to 25%, in steps of 3%. The second column should be the flow rate, in cubic centimeters per second, identified by that percentage. The third column should be the radius implied by this, as computed by Poiseuille's Law. The fourth column should be the percent blockage. (In other words, 0.15 cm radius is a 50% blockage because $0.15/0.3 = 0.5$.)

You can check your work by plugging into the original equation, but you might also find it useful to compare your first table to the following, which is given in increments of 6%.

```
0 % blockage implies  99.9617636500122 cc/minute.
6 % blockage implies  78.0450430095128 cc/minute.
12 % blockage implies  59.9466058383290 cc/minute.
18 % blockage implies  45.1948885141475 cc/minute.
24 % blockage implies  33.3494195216211 cc/minute.
30 % blockage implies  24.0008194523679 cc/minute.
36 % blockage implies  16.7708010049720 cc/minute.
42 % blockage implies  11.3121689849831 cc/minute.
```

```
48 % blockage implies  7.30882030491648 cc/minute.
54 % blockage implies  4.47574398425329 cc/minute.
```

2.3. Industrial Optimization: Shipping Taconite

In this project, you will solve a very simple but realistic shipping route problem, using Linear Programming. This project assumes that you have read Section 4.21 on page 189, about how to do linear programming in Sage.

In particular, a mining conglomerate has seven mines around Minnesota and Wisconsin, and they have shipping facilities at Two Harbors, Green Bay, Minneapolis, plus a shipping complex at the twin ports of Duluth, MN and Superior, WI. You will figure out how much taconite to ship from each mine to each of the four shipping facilities. You have several constraints.

First, it must be the case that the amount of taconite leaving each mine (for all four destinations) should be exactly the amount produced that week. It should not be greater—otherwise you're ordering your mines to ship more taconite than they actually have; it should not be less—otherwise you'll have a build-up of taconite that cannot be stored on site. The total production at each mine is listed in Table 2.1.

Second, the demand and capacity of each shipping facility must be respected. The Two Harbors, WI, facility can receive and ship up to 18,000 tons. The Green Bay facility can receive and ship up to 25,000 tons. The Duluth/Superior shipping complex can receive and ship between 15,000 and 45,000 tons, the minimum being there to ensure there is enough revenue to pay the bills at this expensive location. The Minneapolis facility can receive and ship between 10,000 and 30,000 tons.

Third, the shipping schedule should carry out the task of bringing the taconite to the receiving and shipping stations at minimum cost. The cost of shipping from each mine to each receiving and shipping facility is given in the Table 2.1, in dollars per ton.

TABLE 2.1. The table of shipping costs and mining output.

Shipping Costs

	Duluth, MN Superior, WI	Two Harbors, Minnesota	Green Bay, Wisconsin	Minneapolis, Minnesota	Total Production
Mine 1	$18/ton	$30/ton	$74/ton	$35/ton	8,000 tons/wk
Mine 2	$73/ton	$21/ton	$50/ton	$65/ton	13,000 tons/wk
Mine 3	$37/ton	$93/ton	$67/ton	$61/ton	7,000 tons/wk
Mine 4	$24/ton	$95/ton	$63/ton	$35/ton	16,000 tons/wk
Mine 5	$38/ton	$77/ton	$80/ton	$71/ton	13,000 tons/wk
Mine 6	$73/ton	$15/ton	$39/ton	$68/ton	8,000 tons/wk
Mine 7	$48/ton	$51/ton	$69/ton	$14/ton	9,000 tons/wk

Your goal is to model this problem as a linear program, code it up in Sage, and solve it. To help you out, I offer the following hint. Let $d_1, d_2, \ldots, d_7$ be the amount shipped out of Duluth/Superior from the seven mines. Let $t_1, t_2, \ldots, t_7$ be the amount shipped out of Two Harbors. Let $g_1, g_2, \ldots, g_7$ be the amount shipped out of Green Bay. Let $m_1, m_2, \ldots, m_7$ be the amount shipped out of Minneapolis.

Because you'll be showing a lot of work in this process, I have no fear of giving you the final answer. Here is the output of my program:

```
The minimum cost shipping arrangement is
1883000.0

The quantities shipped to Duluth/Superior from the seven mines are
{1: 8000.0, 2:0.0, 3: 7000.0, 4: 15000.0, 5: 13000.0, 6:0.0, 7:0.0}

The quantities shipped to Two Harbors from the seven mines are
{1: 0.0, 2: 13000.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 5000.0, 7: 0.0}

The quantities shipped to Green Bay from the seven mines are
{1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 3000.0, 7: 0.0}

The quantities shipped to Minneapolis from the seven mines are
{1: 0.0, 2: 0.0, 3: 0.0, 4: 1000.0, 5: 0.0, 6: 0.0, 7: 9000.0}
```

By the way, if you think this problem is complicated, please keep the following in mind. A realistic problem might contain many intermediate collection points, have several dozen mines, and have limited capacities on all of the roads between these points. It is easy to imagine a problem with 30 mines, 12 intermediate collection sites, and 5 ports, for $360 + 60 = 420$ routes. Each route would have a cost and a capacity limit. Each capacity limit is, itself, an inequality. Therefore, these sorts of problems, "in the real world," can get to be enormous.

2.4. Chemistry: Balancing Reactions with Matrices

When you take a class in chemistry, there's lots of fun things to do and learn. One of those things is balancing a chemical equation, which is often easy—however, a few chemical equations are rather hard (for non-chemists) to balance. Here, we'll learn to quickly and easily balance even the most hideous examples by using matrix algebra.

All the examples throughout this project, as well as the original idea of the project itself, are entirely due to Professor Gergely Sirokman of the Wentworth Institute of Technology in Boston, Massachusetts. I thank him for the time he spent crafting these examples for this book.

This project requires knowledge of how to handle linear systems of equations with infinitely many solutions. Appendix D provides complete coverage

of that topic, but some readers will already know about that and will not need to read Appendix D.

2.4.1. Background

Let's start with an energetic example. Back in the age of airships, American airships were often filled with helium. However, Germany had no access to large supplies of helium, and so they used hydrogen instead. This is unfortunate because hydrogen gas can explode in the presence of a spark. The Hindenburg was an airship made by the Zeppelin Company, and it exploded into a fireball in 1937 near Lakehurst, New Jersey. The hydrogen gas combines with oxygen in the air and forms water vapor. Here is the chemical formula for that reaction:

$$2\ H_2 + O_2 \to 2\ H_2O.$$

Let's review what it means for a chemical equation to be balanced. If we have two molecules of H_2 and one molecule of O_2 before the reaction, then we have 4 hydrogen atoms and 2 oxygen atoms before the reaction. Then, if we have 2 molecules of H_2O after the reaction, then we have 4 hydrogen atoms and 2 oxygen atoms after the reaction. No atoms of hydrogen were created or destroyed during the chemical reaction because we had 4 before and 4 after; likewise, no atoms of oxygen were created or destroyed because we have 2 before and 2 after. Since atoms are never created or destroyed in chemical reactions, we would have to reject this chemical equation as unbalanced if we did not have the before-number and the after-number matching. The match must be there for each and every atom in the chemical reaction—if even one before-after pair is unequal, then that means there is a mistake.

The burning of acetylene in a blow torch is a chemical reaction in which acetylene (C_2H_2) and oxygen from the air (O_2) combine to form carbon dioxide (CO_2) and water (H_2O). Based on those chemical formulas, we have to find the positive integers for the following 4 question marks below in order to balance the chemical equation:

$$?\ C_2H_2 + ?\ O_2 \to ?\ CO_2 + ?\ H_2O.$$

Take a moment to try it now yourself, with a little bit of fiddling and some old-fashioned guess and check. Try to find values for the ?'s.

No, really, I mean it—give it a try.

Okay, by this point, I hope you have made an earnest attempt and have found the values to balance the equation:

$$2\ C_2H_2 + 5\ O_2 \to 4\ CO_2 + 2\ H_2O.$$

This was a bit difficult, at least for me. (Perhaps it was easier for you.) With more complex chemical equations, you might be guessing and checking for a while. Unknown to many chemistry students, there is a way to balance chemical equations by using matrices that does not involve any guessing.

2.4.2. Five Cool Examples

Here are five interesting chemical reactions that will serve as our examples.

(1) This reaction comes up in the smelting of ferrous oxide (iron ore) to extract the iron.

$$? \ Fe_2O_3 + ? \ CO \rightarrow ? \ Fe + ? \ CO_2$$

(2) When you put gasoline in your car, octane is one of the important chemicals. This is the reaction when your car burns octane in an abundance of oxygen. If you "rev" the engine up too high, other chemical reactions resulting in CO or even C can occur instead.

$$? \ C_8H_{18} + ? \ O_2 \rightarrow ? \ CO_2 + ? \ H_2O$$

(3) Here is an example of an acid-base reaction, a type of reaction fairly important in chemistry. Calcium hydroxide ($Ca(OH)_2$) has numerous uses, including the treatment of human sewage. Phosphoric acid (H_3PO_4) can be present in human sewage from sources as common as soda drinks. Here we see that calcium hydroxide will treat phosphoric acid and convert it to ordinary water and tricalcium phosphate. As it turns out, tricalcium phosphate is much more benign, as it occurs in animal bones and in cow's milk.

$$? \ Ca(OH)_2 + ? \ H_3PO_4 \rightarrow ? \ Ca_3(PO_4)_2 + ? \ H_2O$$

(4) In the previous example, we said that phosphoric acid (H_3PO_4) occurs in soda drinks. Where does it come from? It can be readily made from pure elemental phosphorous and hypochlorous acid (HClO) in the presence of water. Hydrochloric acid (HCl) is formed as an additional product. Here is the reaction:

$$? \ P_4 + ? \ HClO + ? \ H_2O \rightarrow ? \ H_3PO_4 + ? \ HCl$$

(5) This is one of the preferred methods of extracting gold from diluted rock samples. It is also horrendously toxic (NaCN is sodium cyanide... which is as bad as it sounds!).

$$? \ NaCN + ? \ Au + ? \ O_2 + ? \ H_2O \rightarrow ? \ NaAu(CN)_2 + ? \ NaOH$$

2.4.3. The Slow Way: Deriving the Method

We're now going to slowly develop the matrix method of balancing a chemical equation. A shortcut approach will be given in the next section, but it will make no sense unless you read this derivation very carefully. We will take as our example the middle member of the five examples that we listed moments ago.

$$? \ Ca(OH)_2 + ? \ H_3PO_4 \rightarrow ? \ Ca_3(PO_4)_2 + ? \ H_2O$$

If I had x_1 molecules of $Ca(OH)_2$, x_2 molecules of H_3PO_4, x_3 molecules of $Ca_3(PO_4)_2$, and x_4 molecules of H_2O, then ...

- ... we would have x_1 atoms of calcium on the left and $3x_3$ atoms of calcium on the right;
- ... we would have $2x_1+4x_2$ atoms of oxygen on the left and $8x_3+x_4$ atoms of oxygen on the right;
- ... we would have $2x_1 + 3x_2$ atoms of hydrogen on the left and $2x_4$ atoms of hydrogen on the right;
- ... we would have x_2 atoms of phosphorous on the left and $2x_3$ atoms of phosphorous on the right.

Since, for each element, we need to have the same number of atoms on the left as on the right, then we have the following 4 equations in 4 variables:

$$\begin{aligned} x_1 &= 3x_3, \\ 2x_1 + 4x_2 &= 8x_3 + x_4, \\ 2x_1 + 3x_2 &= 2x_4, \\ x_2 &= 2x_3. \end{aligned}$$

The matrix equivalent to that system of equations would be

$$A = \left[\begin{array}{cccc|c} 1 & 0 & -3 & 0 & 0 \\ 2 & 4 & -8 & -1 & 0 \\ 2 & 3 & 0 & -2 & 0 \\ 0 & 1 & -2 & 0 & 0 \end{array}\right].$$

According to Sage, the RREF is

```
[   1    0    0 -1/2    0]
[   0    1    0 -1/3    0]
[   0    0    1 -1/6    0]
[   0    0    0    0    0]
```

As you can see, we have a row of all zeros. We have a pivot for x_1, x_2, and x_3; thus those variables are determined. There is no pivot for x_4, so that variable is free. The "standard form" of the infinite set of solutions is written as follows:

$$\begin{aligned} x_1 &= (1/2)x_4, \\ x_2 &= (1/3)x_4, \\ x_3 &= (1/6)x_4, \\ x_4 &= x_4. \end{aligned}$$

However, it is customary only to consider integer values for the coefficients of a chemical reaction's equation. What do we do when we want integer-only solutions in a case like this? It is an extremely rare trick, not commonly taught but easy to learn. First, we take the lcm (least common multiple) of all the denominators of all the fractions. In this case, the lcm

of $\{2, 3, 6\}$ is six. Second, we plug in $x_4 = 6n$ where n is a new dummy variable. Now we have

$$\begin{aligned} x_1 &= (1/2)6n = 3n, \\ x_2 &= (1/3)6n = 2n, \\ x_3 &= (1/6)6n = n, \\ x_4 &= 6n. \end{aligned}$$

For any integer n, the four values that you would get form an integer solution to the given equations. This includes nonsense values like $n = 0$ or negative values of n. (Or to be precise, the $n \leq 0$ cases are mathematically valid solutions to the system of equations, but they are not chemically meaningful.) We want the simplest non-zero all-positive solution, so we should plug in $n = 1$. Now we have

$$x_1 = 3, \quad x_2 = 2, \quad x_3 = 1, \quad x_4 = 6,$$

giving the final chemical reaction

$$3\ \mathrm{Ca(OH)_2} + 2\ \mathrm{H_3PO_4} \rightarrow 1\ \mathrm{Ca_3(PO_4)_2} + 6\ \mathrm{H_2O},$$

but we should note that chemists do not usually write the "1" in front of the third molecule.

Now we should check our work. As you can see, ...

- ... we have $3(1)+0 = 3$ atoms of calcium on the left and $1(3)+0 = 3$ atoms of calcium on the right;
- ... we have $3(2)+2(4) = 14$ atoms of oxygen on the left and $1(8)+6(1) = 14$ atoms of oxygen on the right;
- ... we have $3(2) + 2(3) = 12$ atoms of hydrogen on the left and $0 + 6(2) = 12$ atoms of hydrogen on the right;
- ... we have $0 + 2(1) = 2$ atoms of phosphorous on the left and $1(2) + 0 = 2$ atoms of phosphorous on the right.
- All of these are exact matches—therefore we declare success!

2.4.4. Balancing the Quick Way

That was a really excellent but slow process. Now we might want to balance chemical equations faster. The key to remember is "atoms are rows, and molecules are columns." A hint to remembering that is to observe that the word "row" is shorter than the word "column," and similarly the word "atom" is shorter than the word "molecule." Pairing short-to-short and long-to-long, it is natural that atoms correspond to rows and molecules correspond to columns.

The atoms can be encoded in any order. Let's try the second example, the combustion of octane.

$$?\ \mathrm{C_8H_{18}} + ?\ \mathrm{O_2} \rightarrow ?\ \mathrm{CO_2} + ?\ \mathrm{H_2O}$$

Let's go one atom at a time. We're going to ask each molecule how many of the particular atom it has. For the reagents (the left-hand side of the arrow), we will use positive integers; for the products (the right-hand side of the arrow), we will use negative integers.

Carbon: The four molecules have 8, 0, 1, and 0 atoms of carbon; thus we write `8, 0, -1, 0`.

Hydrogen: The four molecules have 18, 0, 0, and 2 atoms of hydrogen; thus we write `18, 0, 0, -2`.

Oxygen: The four molecules have 0, 2, 2, and 1 atoms of oxygen; thus we write `0, 2, -2, -1`.

Now we define the matrix and ask for its RREF, in the list-of-lists notation that we learned about on page 23. However, we must be careful to add the column of zeros as the rightmost column of the matrix, signifying the $= 0$ part of the equations.

```
B= matrix([ [8, 0, -1, 0, 0], [18, 0, 0, -2, 0], [0, 2, -2, -1, 0]])

print "Question:"
print B
print "Answer:"
print B.rref()
```

Now we have the output

```
Question:
[ 8  0 -1  0  0]
[18  0  0 -2  0]
[ 0  2 -2 -1  0]
Answer:
[     1      0      0   -1/9      0]
[     0      1      0 -25/18      0]
[     0      0      1   -8/9      0]
```

Quickly, we verify that the matrix that we entered was the matrix that we had desired to enter. Next, we take the lcm of the denominators in the column of interest. The lcm of $\{9, 18, 9\}$ is clearly 18. For the first three coefficients, we multiply the entries in the informative column by the negative lcm (in this case, -18), and obtain 2, 25, and 16. For the fourth coefficient, it is just the lcm itself, namely 18. We conclude with

$$2\ C_8H_{18} + 25\ O_2 \to 16\ CO_2 + 18\ H_2O$$

but now we have to check it.

- We have $2(8)+0 = 16$ atoms of carbon on the left and $16(1)+0 = 16$ atoms of carbon on the right.
- We have $2(18)+0 = 36$ atoms of hydrogen on the left and $0+18(2) = 36$ atoms of hydrogen on the right.

- We have $0+25(2) = 50$ atoms of oxygen on the left and $16(2)+18 = 50$ atoms of oxygen on the right.
- Since the numbers match, we declare victory!

Last but not least, it would be good to note that chemists have shortcuts that they use which simplify matters significantly, but which require chemical intuition. For example, PO_4 and OH are polyatomic ions, which means that you can "pretend" they are individual atoms. This can remove one variable from the linear system of equations, which is significant.

Your Challenge:

Now try the first, fourth, and fifth examples on your own. Good luck, and don't forget to check your work!

2.5. Physics: Ballistic Projectiles

Ballistic projectile motion problems are a staple in calculus classes as well as physics classes. Here we assume that the reader has knowledge of *Calculus I* and *Physics I*, or the high school equivalents. We're going to explore two relatively easy problems—where there is no need to use Sage—and then one that is a bit more difficult—where Sage or some other computational tool would be required.

For a wide class of simple and intermediate problems, the following seven-step technique often completely solves the problem.

(1) Read the problem very carefully, and write down a sum of forces equation—often with the aid of a free-body diagram. This, in turn, produces an acceleration function using $\sum F = ma$.
(2) Integrate the acceleration function to get the velocity function, which unfortunately will have a $+C$.
(3) Use some information from the problem to compute the value of C.
(4) Integrate the velocity function to get the altitude function, which unfortunately will have a (probably different) $+C$.
(5) Again, use some information from the problem to compute the value of C.
(6) Check that your three functions are consistent with all the given data (by plugging in values) and with each other (by taking derivatives).

Note: At this point, you have all three functions explicitly and can answer almost any inquiry about the projectile.

(7) Reread the question and see what is actually asked for.

Throughout this project, $y(t)$ will represent the altitude of a projectile, and $y'(t) = v(t)$ its velocity. Likewise, $y''(t) = v'(t) = a(t)$ is the projectile's acceleration. We will ignore air resistance in all cases. Due to the perils of rounding error, it is unfortunately necessary that we use $g = 9.80665$ m/s^2 throughout this problem—we really do need all six significant figures.

Before we start, it should be stated that there are two ways of approaching these sorts of problems. Typically, applied mathematicians use lots of digits of precision midway and only round at the end to reflect the uncertainty in the inputs. However, they do not usually have the units inside of equations attached to all of the coefficients and constants. In contrast to this, physicists use units inside of equations (!) attached to all of the coefficients and constants. Furthermore, in the first-year university courses, some textbooks round off numbers stage-by-stage in the middle steps of the computations to reflect the limited precision of the input data. Since I am accustomed to the applied mathematician's technique, I use it here, so as to minimize the chance of a typographical mistake.

You can do these next two warm-up examples with a pencil and a scientific calculator. Then we will proceed to a problem where we need Sage.

2.5.1. Our First Example of Ballistic Trajectories

Suppose an artillery piece is located on a hill, such that the muzzle is 25 m above the surrounding countryside. It fires its projectile at a muzzle velocity of 300 m/s, at an angle of 30° above the horizontal. How far does the projectile go?

First, we have to construct an acceleration function. The only force on the projectile (mid-flight) is the weight of the projectile due to the earth's gravity. We write

$$\begin{aligned} \sum F &= ma(t), \\ -mg &= ma(t), \\ -g &= a(t), \\ a(t) &= -9.80665. \end{aligned}$$

Second, we integrate to find $v(t)$ with a +C:

$$\begin{aligned} \int a(t)\,dt &= \int a(t)\,dt, \\ \int v'(t)\,dt &= \int -9.80665\,dt, \\ v(t) &= -9.80665t + C. \end{aligned}$$

Third, we know that $v(0)$ is the vertical component of the muzzle velocity. That's computed with

$$(300)(\cos 30^\circ) = (300)(0.866025) = 259.807$$

and therefore

$$\begin{aligned} v(t) &= -9.80665t + C, \\ v(0) &= -9.80665(0) + C, \\ 259.807 &= C. \end{aligned}$$

Now we know that $v(t) = -9.80665t + 259.807$, at least during flight. Fourth, we integrate to find $y(t)$ with a +C:

$$\begin{aligned}
\int v(t)\,dt &= \int v(t)\,dt, \\
\int y'(t)\,dt &= \int (-9.80665t + 259.807)\,dt, \\
y(t) &= -\frac{1}{2}(9.80665)t^2 + 259.807t + C, \\
y(t) &= -4.90332t^2 + 259.807t + C.
\end{aligned}$$

Fifth, we plug in the fact that $y(0)$ is the altitude of the hill, or 25 m:

$$\begin{aligned}
y(t) &= -4.90332t^2 + 259.807t + C, \\
y(0) &= -4.90332(0)^2 + 259.807(0) + C, \\
25 &= C.
\end{aligned}$$

Finally, we have

$$y(t) = -4.90332t^2 + 259.807t + 25$$

and we can begin the most important part of the problem: checking our work.

- We can check that $y'(t) = -9.80664t + 259.807$, which matches our $v(t)$ except for rounding error.
- We can check that $y''(t) = -9.80664$, which matches our $a(t)$ except for rounding error.
- We can check that $y'(0) = v(0) = 259.807$, as desired. (That's the vertical component of the muzzle velocity.)
- We can check that $y(0) = 25$, as desired. That's the altitude of the artillery piece's muzzle.

Now, we should compute what the question actually wants to know. It wants to know the range of the projectile (how far it goes) so we have to calculate the time of flight and multiply that by the horizontal velocity. To find the time of flight, we just have to find out when the projectile hits the ground (when flight ends) because flight begins when $t = 0$. Of course, when the projectile hits the ground, the altitude is zero. Therefore, we set $y(t) = 0$ and use the quadratic formula:

$$\begin{aligned}
0 &= y(t), \\
0 &= -4.90332t^2 + 259.807t + 25, \\
t &= \frac{-259.807 \pm \sqrt{(259.807)^2 - 4(-4.90332)(25)}}{2(-4.90332)}, \\
t &= -0.0960512 \text{ or } 53.0820.
\end{aligned}$$

We check our work with $y(53.0820) = 0.0174218$ m, a difference of 17 mm, which is not a problem, but which is obviously mere rounding error. Next, we need the horizontal velocity

$$(300)(\sin 30^\circ) = (300)(1/2) = 150$$

and finally the range of the projectile is

$$(150)(53.0820) = 7{,}962.30.$$

Since a mile is 1,609 meters, a range of 7,962.30 m seems plausible. After all, the purpose of artillery is to strike targets several miles away. We should realize, however, that our altitude of the hill being 25 m is a measurement accurate probably to the nearest meter or half meter, and not to six significant digits. The muzzle velocity is probably also only known to the nearest 1%. Therefore, it is better to report the answer to two significant figures and reply that the projectile's range is 7,900–8,000 meters.

2.5.2. Our Second Example of Ballistic Trajectories

Now we're going to explore an air-defense problem. We're going to simulate the computer of an air-defense radar system that is tracking incoming warheads. This is not an unrealistic problem. During the First Gulf War (1991), Iraq fired SS-1 "Scud" missiles over long distances to Israel and Saudi Arabia. Some could be shot down during flight, by surface-to-air missiles, but there were many incoming Scuds. It is useful to compute the point of impact—after all, if a warhead is heading toward the empty desert, then that's fine, but if a warhead is heading toward a shopping mall, then that's an emergency. Just as this book was going to the publisher in July of 2014, similar computations were being done by the Israeli air-defense system "Iron Dome," so rest assured that these techniques are rather relevant.

To keep the numbers and the model simple, we'll work with a problem about artillery shells, and not about regional ballistic missiles.

A projectile has been detected in the vicinity of the air-defense battery for which our computer has responsibility. The radar's computer has defined its coordinate system so that the location of the radar is $(0, 0, 0)$. A radar snapshot has been taken of the incoming projectile, reporting its position and velocity. The y direction is straight up, the x direction is due east, and the z direction is due south. The radar has reported that the projectile is at a position of (12,000; 5,000; 7,000) meters, with a velocity of $(-60, -240, 80)$ in m/s. The location of impact must first be computed. For simplicity, let $t = 0$ be the time of the snapshot, and not the time of launch.

First, we should use the seven-step method to find $y(t)$, the altitude function. I'll allow you to do that yourself, so that you can practice. However, I'll give you the answers after each step, so that you can check your work.

(1) Step One: $a(t) = -9.80665$.
(2) Step Two: $v(t) = -9.80665t + C$.

(3) Step Three: $v(t) = -9.80665t - 240$.
(4) Step Four: $y(t) = -4.90332t^2 - 240t + C$.
(5) Step Five: $y(t) = -4.90332t^2 - 240t + 5{,}000$.
(6) Step Six:
- The first derivative is $y'(t) = -9.80664t - 240$, matching $v(t)$.
- The second derivative is $y''(t) = -9.80664$, matching $y(t)$.
- The altitude at $t = 0$ is $y(0) = 5{,}000$, as desired.
- The altitude at $t = 0$ is $y'(0) = -240$, as desired.

(7) Step Seven: We need to find the impact time, just as in the last problem. However, this does not happen for every problem. In any case, the time of impact is somewhat soon, at $t = 15.7593$ seconds.
(8) Step Seven, Continued: At a rate of $\dot{x} = -60$ m/s, in 2.00148 seconds, the projectile will have moved -945.558 meters in the x direction; at a rate of $\dot{z} = +80$ m/s, the projectile will have moved 1,260.74 meters in the z direction. Of course, at impact, $y = 0$.
(9) Therefore the point of impact is $(-11{,}054.4; 0; 8{,}260.74)$.

Note: The symbols $\dot{x}$ and $\dot{z}$ are just abbreviations for dx/dt and dz/dt.

2.5.3. Counter-Battery Fire

In real life, one of the fun things that can be done is to mathematically trace the parabola of flight backward to detect the launch point. Once the point of launch is known, clearly there is an artillery piece there, and therefore it would be prudent to heavily bombard that location to destroy the enemy artillery battery. This technique, called "counter-battery fire," was developed in the First World War and has been a mainstay of modern warfare since that time. Another reason the technique is fun is that you use "the wrong value" of t when solving $y(t) = 0$. Namely, you use the root with $t < 0$, something which almost never happens in projectile motion problems.

The time of launch is when $y(t) = 0$ but $t < 0$. (Actually, we're tacitly assuming that the altitude of the launcher, the observer, and the impact are all equal or nearly equal. However, if they were not equal, the adjustments would be straightforward.) The quadratic equation reveals to us that in this case, from the previous subsection $y(t) = -4.90332t^2 - 240t + 5{,}000$ and therefore the time of impact is $t = -64.7057$ seconds. This sounds reasonable, as it makes the total time of flight 80.4650 seconds, being a bit more than one minute.

The coordinates of the firing howitzer can be obtained by computing

$$\begin{aligned} (-60)(-64.7057) + 12{,}000 &= 15{,}882.3, \\ (+80)(-64.7057) + 7{,}000 &= 12{,}176.4. \end{aligned}$$

Therefore, we report that the enemy artillery battery is at the following coordinates:

$$(15{,}882.3; \quad 0; \quad 12{,}176.4)$$

The Truth about Air Resistance:
Of course, we've neglected wind and air resistance—in real life, it would be better to include them. You're probably curious about how this is actually done. What happens is that time is divided into tiny intervals, around 10^{-4} or 10^{-5} seconds, and during those time slices, one pretends that the effect of wind and air resistance is a constant function or a linear function. About 10^6 or 10^7 time slices represent the entire flight path of the projectile. Using techniques that you will learn in the first parts of Chapter 5, it is extremely easy to write a computer program which repeatedly steps forward in time by those tiny time slices.

The modification by a constant function or linear function is going to only slightly elevate the complexity of the calculation for each time slice. However, because the time slices are so tiny, the accuracy of the overall computation can be very high. The general concept we are discussing here is called "numerical simulation," but we are unable to explore this alternative approach any further.

2.5.4. Your Challenge: A Multi-Stage Rocket

Now that you have practiced the techniques of projectile motion, let's consider the test flight of a rocket. In this case, the rocket malfunctions and crashes, and we are going to model its flight path.

During normal operations, the engine provides a constant force of thrust of 1.27 meganewtons. However, eleven seconds into flight, the fuel pump control computer malfunctioned and the flow of fuel to the engine was slowly throttled to zero. Of course, the rocket crashed. A good model of the thrust of the engine after the malfunction would be

$$f(t) = 1.27e^{(11-t)/2} \quad \text{for all } t > 11,$$

again measured in meganewtons. It will be useful to know that the mass of the rocket at launch was 107,000 kilograms.

We have two simplifying assumptions. While fuel is clearly being burned during the first eleven seconds, it is okay to pretend as though the mass is constant for the entire problem. (That's because only a tiny fraction of the fuel was burned during the flight.) Second, the rocket does not get very high at all, and therefore it is okay to pretend $g = 9.80665$ m/s^2 is a constant. In reality, g does change when you reach higher and higher altitudes, but a projectile has to get relatively high before that matters too much.

Let's begin. First, find the altitude function when the engine was working normally—namely $0 < t < 11$. You can assume that the rocket was sitting on the launch pad, at $y = 0$, for a long time prior to launch. Once you have this, compute the altitude, velocity, and acceleration 11 seconds after launch—the moment that the malfunction occurred.

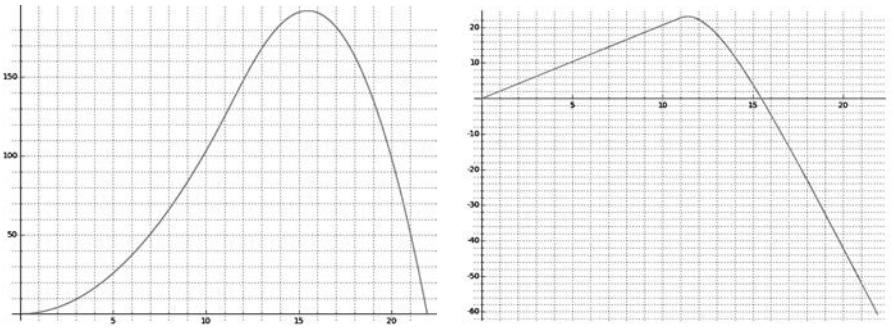


FIGURE 2.2. The graphs for the physics project.

Next, using the function $f(t)$ above, compute a new acceleration function, a new velocity function, and a new position function, for use after the malfunction. That will enable us to find the time of impact.

There are ten deliverable items for this project:

- First, second, and third, provide the acceleration, velocity, and position functions before the malfunction.
- Fourth, fifth, and sixth, provide the acceleration, velocity, and position functions after the malfunction.
- Seventh, provide the time of impact.
- Eight, provide a single graph showing the altitude for the entire flight, from launch to impact.
- Ninth, provide a single graph showing the velocity for the entire flight, from launch to impact.
- Tenth, provide the maximum height achieved.

To help you understand if you've done the problem correctly, the two graphs are given in Figure 2.2.

2.6. Cryptology: Pollard's $p-1$ Attack on RSA

The RSA Cryptosystem is easily the most famous of cryptographic systems in use today. Nearly every e-commerce transaction on the Internet uses TLS (Transport Layer Security) or SSL (the Secure Sockets Layer) and therefore uses RSA to exchange block-cipher keys. The security of RSA is based upon the difficulty of factoring the product of two (enormous) prime numbers.

This project is one of several ways of attacking the RSA Cryptosystem. The project assumes familiarity with RSA and a bit of number theory. If the reader is unfamiliar with RSA, then this project cannot be attempted. However, I would encourage such a reader to learn RSA. It is relatively easy to learn, at least compared to other mathematical concepts, and it provides a window into the practical applications of otherwise "pure" topics such as prime numbers and modular arithmetic.

This project closely follows the notation given in *Cryptography and Coding Theory*, Second Edition, by Wade Trappe and Lawrence Washington, published by Pearson in 2005. See Ch 6.4 of that textbook. The original method was discovered by John Pollard in 1974.

Alternatively, Minh Van Nguyen has written a tutorial for Sage and cryptography that uses RSA as its example. The title is "Number Theory and the RSA Public Key Cryptosystem" and you can find that tutorial at

`http://www.sagemath.org/doc/thematic_tutorials/numtheory_rsa.html`

Last but not least, this project requires some knowledge of `for` loops, as covered in Chapter 5 of this book.

2.6.1. Background: B-Smooth Numbers and $B!$

Number theorists will often talk about certain positive integers as being "smooth." This is a vague term meaning that the number has no large prime factors, but rather that the prime factorization is entirely composed of small primes. More precisely, for any even number k, we say that a number is k-smooth if all of its prime factors are less than k. For example, 15, 10, 100, and $2^{5,000}$ are all 6-smooth, but 14 is not because $14 = 2 \times 7$ and 7 is larger than 6. As another example, 14 and 1,400 are 8-smooth, but 11 is not 8-smooth. This notion can be interesting in several branches of number theory, but it is especially of interest in cryptography. This is the standard way of defining what it means for a number to be "smooth."

It so happens that an RSA public key can easily be broken (by factoring its N) if either $p-1$ or $q-1$ are smooth. That's a rather weird statement, of course. First of all, p and q are not public information. However, that's no barrier to this attack because it is only required that $p-1$ or $q-1$ be smooth—we do not need to know $p-1$ or $q-1$ or even which of the two primes is "one plus a smooth number."

Another oddity is that we're going to use a non-standard definition of smoothness. We're going to pick a large number B and ask if either p or q divides $B!$. Shortly, we'll use $B = 10{,}000$, but for now, we'll use $B = 100$. Let's explore how this usage of 100! compares to a number being 100-smooth.

The only way that a number z fails to be 100-smooth is if z has a prime in its prime factorization that is greater than 100, such as perhaps 101 or 103. Since 101 and 103 are prime and since both are greater than 100, we know that neither 101 nor 103 will divide any of the numbers from 1 to 100. Therefore, by the famous number theory lemma that $p|ab$ if and only if $p|a$ or $p|b$, we can conclude that 101 does not divide 100!, and likewise 103 does not divide 100!. Likewise, any multiple of 101 or 103 will also not divide 100!. The same goes for any prime that is larger, such as 107 and so forth. Therefore, any number which has a prime in its factorization that is greater than 100 will definitely not divide 100!.

However, we might be interested in the converse. If a number y does not divide 100!, is it always the case that y is not 100-smooth? Sadly, here we will encounter some exceptions. Using the commands

```
factor( factorial( 100 ) )
```

we learn that

$$\begin{aligned} 100! &= (2)^{97}(3)^{48}(5)^{24}(7)^{16}(11)^{9}(13)^{7}(17)^{5}(19)^{5}(23)^{4}(29)^{3}(31)^{3}(37)^{2} \\ &\quad \cdots (41)^{2}(43)^{2}(47)^{2}(53)(59)(61)(67)(71)(73)(79)(83)(89)(97). \end{aligned}$$

Therefore, you can see that 2^{98}, 3^{49}, and 5^{25} will not divide 100!. However, those three numbers are all 6-smooth, so they are surely 100-smooth. So we have now located three examples of 100-smooth numbers that do not divide 100!.

In fact, we've now demonstrated that requiring numbers to be divisors of 100! is *a stricter standard* than requiring them to be 100-smooth. In general, the set of numbers dividing $B!$ is a subset of the numbers that are B-smooth. Overall though, if you consider numbers less than $B!$, the exceptions are few and far between, and the concepts are roughly equivalent.

2.6.2. The Theory behind the Attack

We need one more theorem from number theory. For any prime number p and for any $a \not\equiv 0 \bmod p$, Fermat's Little Theorem says that

$$a^p \equiv a \bmod p \qquad \text{or equivalently} \qquad a^{p-1} \equiv 1 \bmod p.$$

Suppose $p-1$ divides $B!$. That means that there is some k such that $(k)(p-1) = B!$. Then we can conclude

$$2^{B!} = 2^{(k)(p-1)} = (2^{p-1})^k \equiv 1^k \equiv 1 \bmod p.$$

Because $2^{B!} \equiv 1 \bmod p$, it must be the case that

$$(2^{B!} - 1) \equiv 0 \bmod p$$

or, more plainly, that $c = (2^{B!} - 1)$ is a multiple of p. This key fact, that c is a multiple of p, is the heart of the entire idea.

In fact, on further thought, it must be the case that both c is a multiple of p in the ordinary integers and that the image of c mod N is a multiple of p in "the integers mod N." Most readers will take this rather technical point on faith and can skip to the start of the next paragraph. For those who want absolute rigor, observe that if c is a multiple of p, then $c = jp$, for some integer j. We have no reason to believe that j is divisible by q, but let's divide by q anyway, getting quotient j_1 and remainder j_2 with $0 \leq j_2 < q$ and $j_1 \geq 0$. This means that $j = qj_1 + j_2$ and thus

$$c = jp = (qj_1 + j_2)p = pqj_1 + pj_2 = Nj_1 + pj_2 \equiv 0 + pj_2 \bmod N,$$

which allows us to conclude that the image of c mod N is just pj_2, clearly a multiple of p.

What have we gained by this? Well, since $N = pq$, we know that N is a multiple of p as well. Since both c and N are multiples of p, then $\gcd(c, N)$ is also a multiple of p. That gcd can be only one of two values, actually—that gcd is either p or N.

Therefore, we first compute $c = 2^{B!} - 1 \bmod N$, which is no minor task, and then compute the value of $\gcd(c, N)$. If we get something between 1 and N, we know it must be p, and ordinary division will provide us with $q = N/p$. We have factored N and we have broken this person's public key.

On the other hand, if we get that the gcd is N, we are unlucky and have[3] failed. The gcd will be N (resulting in a failure of the attack) if and only if q also divides c. That will essentially only occur if both $p - 1$ and $q - 1$ divide $B!$.

It so happens that having one of *either* $p-1$ or $q-1$ divide $B!$ is somewhat rare. However, having *both* $p - 1$ and $q - 1$ dividing $B!$ is extremely rare. More precisely, p and q are supposed to be generated randomly, and therefore the probability of $p-1$ dividing $B!$ is independent of the probability of $q-1$ dividing $B!$. The probability of both of them dividing $B!$ is therefore *the square of the probability* of one of them happening to divide $B!$.

Accordingly, we fully expect that the attack will work for any N, so long as either $p - 1$ or $q - 1$ divides $B!$.

2.6.3. Computing $2^{(B!)} \bmod N$

Let's reflect upon the fact that computing

$$2^{(10{,}000!)} \bmod N$$

is not a task that we are asked to do very often. Therefore it would be unwise to dive into the problem without some thought at first. We might ask ourselves, exactly how large is 10,000! going to be? Or, more precisely, how many digits will it have?

It turns out that this question is easy to answer if we recall Stirling's approximation:

$$B! \approx B^B e^{-B} \sqrt{2\pi B}$$

[3]If this ever happens to you in practice, try again with a smaller B. If the largest prime in the prime factorization of $p - 1$ is f_1 and the largest prime in the prime factorization of $q - 1$ is f_2, then choosing a B between f_1 and f_2, presumably by guess and check, will result in a successful outcome—namely $\gcd(c, N) = p$.

and proceed by taking the common logarithm

$$\begin{aligned}
\log_{10}(B!) &\approx \log_{10}\left(B^B\right)\left(e^{-B}\right)\left(\sqrt{2\pi B}\right) \\
&\approx \left(\log_{10} B^B\right) + \left(\log_{10} e^{-B}\right) + \left(\log_{10}\sqrt{2\pi B}\right) \\
&\approx (B\log_{10} B) - (B\log_{10} e) + \frac{1}{2}\left(\log_{10} 2\pi B\right) \\
&\approx B\left(\log_{10} B - \log_{10} e\right) + \frac{1}{2}\left(\log_{10} 2\pi\right) + \frac{1}{2}\left(\log_{10} B\right) \\
&\approx B\left(\log_{10} B - 0.434294481\ldots\right) + 0.399089934\ldots + \frac{1}{2}\log_{10} B \\
&\quad \ldots \text{or in our case of } B = 10{,}000 \text{ and } \log_{10} B = 4\ldots \\
&\approx 10{,}000\left(4 - 0.434294481\ldots\right) + 0.399089934\ldots + \frac{1}{2}(4) \\
&\approx 35,657.0551\ldots + 2.39908993\ldots \\
&\approx 35,659.4542\ldots.
\end{aligned}$$

Having come to realize that 10,000! is a number that has 36,660 digits (approximately), we probably should not compute it and then ask for $2^{(10{,}000!)}$. That might take a very long time to compute. We have to find a different approach.

Consider computing

$$\begin{aligned}
c_1 &= 2^1 \bmod N, \\
c_2 &= (c_1)^2 \bmod N, \\
c_3 &= (c_2)^3 \bmod N, \\
c_4 &= (c_3)^4 \bmod N, \\
c_5 &= (c_4)^5 \bmod N, \\
c_6 &= (c_5)^6 \bmod N, \\
&\vdots \quad \vdots \quad \vdots
\end{aligned}$$

Then we would have

$$c_6 \equiv (c_5)^6 \equiv ((c_4)^5)^6 \equiv (((c_3)^4)^5)^6 \equiv ((((c_2)^3)^4)^5)^6 \equiv (((((c_1)^2)^3)^4)^5)^6 \equiv 2^{(1)(2)(3)(4)(5)(6)}$$

but observe that that's just what we wanted to compute, namely

$$2^{(1)(2)(3)(4)(5)(6)} \equiv 2^{(6!)}.$$

In a similar manner, we can see that $c_B \equiv 2^{(B!)} \bmod N$.

2.6.4. Your Challenge: Make All This Happen in Sage

Your task is an easy one. Using the strategy above, compute

$$c = 2^{(10,000!)} \bmod N.$$

Next, compute $p = \gcd(c, N)$, followed by $q = N/p$, and check your work by multiplying p and q back together again. Here are seven test cases for you:

3571770308958050087813192668766411035818396781514951351291330670101 27

5000272002251811540446579586816039346308785565641862331905860763123733

1112930940347693048841670514581743208135955104481382748661520020673659 27

1891149894297520829262854575513696427873817900392608023074521104903045 47

2507336544824685689216200428668597188529200280260765471779747582391091 77

2015573899005400956135592195412995405224052593293997368248582528763765213110530067105771630572340 93

8620215476436315823969982122087229142882586442347913079505829164427472220397956094177419322783171 21

By the way, you should be able to cut-and-paste these numbers into Sage from the `pdf` file of this book, available on my webpage[4]. You do not need to retype these enormous numbers!

2.6.5. Safeguards against Pollard's Factorial Attack

As you can see, RSA public keys can be broken (that is to say, N can be factored) if either $p-1$ or $q-1$ is smooth. A smooth number (for example, a 6-smooth or 100-smooth number) has a rather large number of prime factors which each contribute a small magnitude toward the final product. For example,

$$10^{100} = 2^{100}5^{100}$$

is a very smooth number, with 200 primes in its prime factorization (counting multiplicities) and none contribute more than a paltry 5 to the final, enormous, product. What is the opposite extreme to this?

The other extreme would be a prime number. There is only one prime factor, and it makes all the contribution to the magnitude by itself. Of course, p and q are odd and thus neither $p-1$ nor $q-1$ can be prime. However, they could be 2 times some prime number. Prime numbers p where both p and $(p-1)/2$ are prime simultaneously are called "safe primes." There is an almost identical classification, called[5] the "Sophie Germain primes;" these are p such that both p and $2p+1$ are prime.

To make Pollard's attack totally infeasible, many RSA implementations require both primes to be "safe primes." For these numbers, B would have to be $(p-1)/2$ or greater. Even if p is only 200 bits long, which is far too small, then $(p-1)/2$ will be 199 bits long, and that would make it

[4] `www.gregorybard.com`

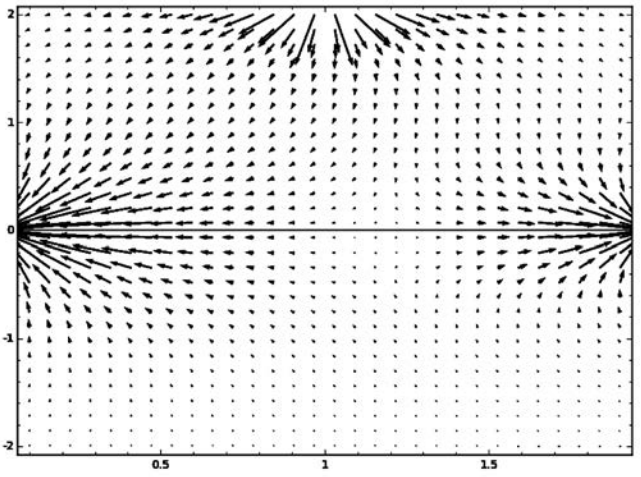
[5] Named for Marie-Sophie Germain (1776–1831), one of the earliest women to be recognized for mathematical achievement by the Paris Academy of Sciences. She is most noted for progress toward the solution of Fermat's Last Theorem, where the Sophie Germain primes played a significant role in her lemmas.

computationally infeasible to compute $2^{B!}$. The sun would run out of fuel, turning into a red giant[6] long before such a computation could terminate.

2.7. Mini-Project on Electric Field Vector Plots

Take a moment to read Section 3.7, Vector Field Plots, beginning on page 116, if you haven't already read it. In this mini-project, you're going to be asked to modify the model that I used in the subsection "An Application from Physics" to generate the vector field plot of three electric charges. We'll be adding a fourth electric charge, with $q_3 = -1$, and it is glued to the point $(1, 2.25)$.

Your task is to modify my code to accommodate this fourth electric charge and then produce the vector field plot. You should obtain the following image:



[6]This will happen roughly 5 billion years from now.