6.055J / 2.038J The Art of Approximation in Science and Engineering
Spring 2008

6.055J/2.038J (Spring 2008)

Solution set 2

Do the following warmups and problems. Due in class on ***Monday, 03 Mar 2008****.*

Open universe: *Collaboration, notes, and other sources of information are* ***encouraged.*** *However, avoid looking up answers until you solve the problem (or have tried hard). That policy helps you learn the most from the problems.*

Bring a photocopy to class on the due date*, trade it for a solution set, and figure out or ask me about any confusing points. Your work will be graded lightly: P (made a reasonable effort), D (did not make a reasonable effort), or F (did not turn in).*

Warmups

1. Fish tank

Estimate the mass of a typical home fish tank (filled with water and fish): a useful exercise before you help a friend move who has a fish tank.

By having good aim or getting lucky, I won two tiny goldfish at our elementary school's annual fair. They lived happily for many years in our fish tank, eventually growing to 7 inches in length. The tank was perhaps 1 foot wide, 3 feet long, and 1.5 feet high, which is a volume of

$$V \sim 0.3\,\mathrm{m} \times 1\,\mathrm{m} \times 0.5\,\mathrm{m} \sim 0.15\,\mathrm{m}^3.$$

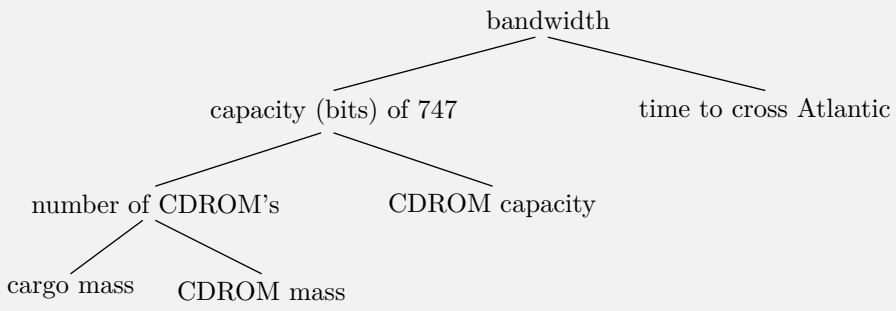
If it filled two-thirds with water, that's $0.1\,\mathrm{m}^3$ of water or 100 kg! The glass tank itself has a much lower mass so the main contribution is from the water.

I estimated the mass to only one digit, neglecting the mass of the empty tank. That's no worse than the errors from the length estimates. The tank itself is an old memory (from 30 years ago) and there's no need to overengineer the estimate.

2. Bandwidth

Estimate the bandwidth (bits/s) of a 747 crossing the Atlantic filled with CDROM's.

Divide and conquer! Here's a tree on which to fill values:



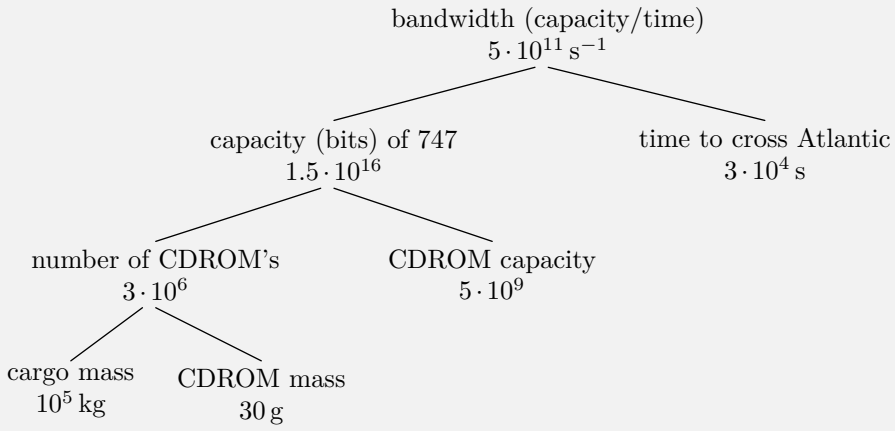
First I estimate the cargo mass. A 747 can easily carry about 400 people, each person having a mass (with luggage) of, say 140 kg. The total mass is

$$m \sim 400 \times 140\,\text{kg} \sim 6 \cdot 10^4\,\text{kg}.$$

A special cargo plane, with no seats or other frills for passengers, probably can carry 10^5 kg.

Here are the other estimates. A CDROM's mass is perhaps one ounce or 30 g. So the number of CDROM's is $3 \cdot 10^6$. The capacity of a CDROM is 600 MB or about $5 \cdot 10^9$ bits. The time to cross the Atlantic is about 8 hours or $3 \cdot 10^4$ s.

Now propagate the values toward the root of the tree:



The bandwidth is 0.5 terabits per second.

Despite the large bandwidth offered by a 747 carrying CDROM's (not to mention DVDROM's), trans-Atlantic Internet connections go via undersea fiber-optic cables. Low latency is important!

3. Integrals

Evaluate these definite integrals:

a. $\displaystyle\int_{-10}^{10} x^3 e^{-x^2}\,dx$

The integrand $x^3 e^{-x^2}$ is antisymmetric: Replacing x by $-x$ changes the function's sign. Therefore integrating it over a symmetric range such as -10 to 10 produces zero.

b. $\displaystyle\int_{-\infty}^{\infty} \frac{x^3}{1 + 7x^2 + 18x^8}\,dx$

This integrand is also antisymmetric, so integrating it over a symmetric range such as $-\infty$ to ∞ produces zero.

[As a physics undergraduate, I spent many hours with the table of integrals that we knew affectionately as Gradshteyn. The table was so massive and complete that when we could not locate an integral in it, we suspected that the integral should be zero and went looking for the symmetry.]

Problems

4. Number sum

Use symmetry to find the sum of the integers between 200 and 300 (inclusive).

Reversing the order of the terms is the symmetry operation because the sum is the same in reverse:

$$200 + 201 + 202 + \cdots + 300 = 300 + 299 + 298 + \cdots + 200.$$

Add these sums as follows:

$$\begin{aligned} &200 + 201 + 202 + \cdots + 300 \\ &+ 300 + 299 + 298 + \cdots + 200 \\ \hline &= 500 + 500 + 500 + \cdots + 500. \end{aligned}$$

There are 101 copies of the 500, so this duplicated sum is $500 \times 101 = 50500$. The original sum is one-half of the duplicated sum, so it is 25250.

A quick confirmation is the following Unix pipeline:

```
seq 200 300 | awk 'BEGIN {total=0}; {total += $1}; END {print total};'
```

which produces 25250.

5. Repainting MIT

Estimate the cost to repaint all indoor walls in the main MIT classroom buildings. [with thanks to D. Zurovcik]

Painting requires paint and labor. Let's estimate the ratio of labor to paint costs. A painter charges say \$35/hour. A gallon of paint costs maybe \$25. How much wall or ceiling area gets painted with that much paint? One method is to guess that paint goes onto walls as thickly as a sheet of paper (0.01 cm). So a gallon, which is roughly 4ℓ, can cover

$$\frac{4000\,\text{cm}^3}{0.01\,\text{cm}} \sim 4 \cdot 10^5\,\text{cm}^2 \sim 40\,\text{m}^2.$$

Since walls are roughly 4 m high, that's 10 m of wall, or one wall of a small classroom (or a medium-sized room in a house). It will take probably a couple hours to paint the wall, costing about \$70, so labor is more important than materials in the total cost.

The estimate in the last paragraph is $20\,\text{m}^2$ per hour, or about $\$2/\,\text{m}^2$. Now estimate the classroom area. For a typical classroom, the wall and ceiling area is perhaps five times the area of the wall that separates the classroom from the corridor. So I'll estimate the wall area of the corridors. Walking the Infinite Corridor takes me about 3–4 minutes. Typical walking is 3 miles per hour, which is $1.5\,\text{m}\,\text{s}^{-1}$, makes its length 300 m. There are four storeys, maybe five if you count the basement. So that makes 1500 m of corridor. Multiply by a factor of 2 since there are walls on both sides; by a factor of 2 to account for the other parts of the main building; and by another factor of 2 to account for the other classroom buildings. That gives 10^4 m of wall, or about $4 \cdot 10^4\,\text{m}^2$ of wall. Multiplying by the factor of 5 gives $2 \cdot 10^5\,\text{m}^2$ of surface to be painted.

To paint $2 \cdot 10^5\,\text{m}^2$ would cost about $\$4 \cdot 10^5$.

6. Raindrop speed

a. How does a raindrop's terminal velocity v depend on the raindrop's radius r?

The weight of the raindrop is the density times the volume times g:

$$W \sim \rho r^3 g,$$

where I neglect dimensionless factors such as $4\pi/3$.

At terminal velocity, the weight equals the drag. The drag is

$$F \sim \rho_{\text{air}} v^2 A \sim \rho_{\text{air}} v^2 r^2.$$

Equating the weight to the drag gives an equation for v and r:

$$\rho_{\text{air}} v^2 r^2 \sim \rho r^3 g,$$

so $v \propto r^{1/2}$.

Bigger raindrops fall faster but – because of the square root – not much faster.

b. Estimate the terminal speed for a typical raindrop.

With the g and the densities, the terminal velocity is

$$v \sim \sqrt{\frac{\rho}{\rho_{\text{air}}} gr}.$$

A typical raindrop has a diameter of maybe 6 mm, so $r \sim 3$ mm. Since the density ratio between water and air is roughly 1000,

$$v \sim \sqrt{1000 \times 10\,\text{m}\,\text{s}^{-2} \times 3 \cdot 10^{-3}\,\text{m}} \sim 5\,\text{m}\,\text{s}^{-1}.$$

c. How could you check your estimate in part (b)?

First convert the speed into a more familiar value: 11 mph (miles per hour). If one drives at a speed v_{car}, then raindrops appear to move at an angle $\arctan(v_{\text{car}}/v)$. When $v_{\text{car}} = v$, the drops come at a 45-degree angle. So one way to measure the terminal speed is to drive in a rainstorm, slowly accelerating while the passenger says when the drops come at a 45-degree angle.

You could also run in a rainstorm and note the speed at which a small umbrella has to be at 45 degrees to keep you perfectly dry.

7. Mountains

Look up the height of the tallest mountain on earth, Mars, and Venus, and explain any pattern in the three values.

The heights are:

- Mars: 27 km (Mount Olympus)
- earth: 9 km (Mount Everest)
- Venus: 11 km (Maxwell Montes)

One pattern is that the large planets (earth and Venus) have short mountains, at least short compared to Mount Olympus at a huge height of 27 km.

Large planets presumably have stronger gravitational fields at their surface, which keeps the mountains closer to the ground. The derivation in lecture on mountain heights dropped the dependence on g because we looked only at mountains on earth. Here's the same derivation but retaining g. The weight of a mountain of size l is $W \propto gl^3$, so the pressure at the base is $p \propto gl^3/l^2 \sim gl$. When the pressure exceeds the maximum pressure that rock can support, the mountain can no longer grow upward. So the maximum height l depends inversely on g:

$$l \propto g^{-1}.$$

To test that analysis, here are the gravitational field strengths on the three planets:

- Mars: $3.7\,\mathrm{m\,s^{-2}}$
- earth: $10\,\mathrm{m\,s^{-2}}$
- Venus: $8.9\,\mathrm{m\,s^{-2}}$

The product gl for each planet should be the same, and it roughly is:

- Mars: $10^5\,\mathrm{m^2\,s^{-2}}$
- earth: $0.9 \cdot 10^5\,\mathrm{m^2\,s^{-2}}$
- Venus: $0.98 \cdot 10^5\,\mathrm{m^2\,s^{-2}}$

Fun question: Why aren't mountains on the moon 60 km tall (since the Moon's surface gravity is about one-sixth of earth's surface gravity)?

8. Your turn to create

Invent – but do not solve! – a question for which proportional reasoning or symmetry would help solve.

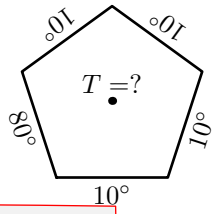
Particularly enjoyable questions might appear on the website or as examples in lecture or the notes (let me know if you do *not* want your name attributed in case your question gets selected).

Optional

9. Heat equation

In lecture we used symmetry to argue that the temperature at the center of the metal sheet is the average of the temperatures of the sides.

Check this result by making a simulation or, if you are bold but crazy, by finding an analytic solution of the heat equation.



This simulation in Python was written to be clear though not necessarily efficient. It uses a lattice to approximate the continuous sheet, and implements so-called relaxation: At each step, the temperature at each point is replaced by the average temperature of the neighbors. The main complications are:

1. The edges of the pentagon are held at fixed temperatures (10 degrees for four edges and 80 degrees for the fifth edge). However, the relaxation step does not maintain those fixed values. So they are re-imposed after each sweep through the lattice.

2. Only one of the edges lies along a coordinate direction. The other four edges have funny slopes, and need to be rasterized. It is the identical problem to rendering lines on a laser printer: Which pixels get the toner? Bresenham's algorithm does the rasterization.

```
# Relaxation simulation of the temperature at the center of the pentagon.
# Four edges are held at 10 degrees, and the fifth at 80 degrees.

from scipy import *

# rounds to nearest integer, and returns an integer
def intround(f):
    return int(round(f))

# Bresenham's algorithm: returns list of lattice points on the line connecting
# r1 and r2
def line(r1, r2):
    x0, y0 = intround(r1[0]), intround(r1[1])
    x1, y1 = intround(r2[0]), intround(r2[1])
    points = []
    steep = abs(y1 - y0) > abs(x1 - x0)
    if steep:
        x0,y0 = y0,x0
        x1,y1 = y1,x1
    if x0 > x1:
        x0,x1=x1,x0
        y0,y1=y1,y0
    deltax = x1 - x0
    deltay = abs(y1 - y0)
    error = 0
    deltaerr = float(deltay) / deltax
    y = y0
    if y0 < y1:
        ystep = 1
    else:
        ystep = -1
```

```
    for x in range(x0,x1+1):
        if steep:
            points.append((y,x))
        else:
            points.append((x,y))
        error += deltaerr
        if error >= 0.5:
            y += ystep
            error -= 1.0
    return points

def complex2pair(c):
    return (real(c),imag(c))

def set_edge_temps(grid):
    for e in lo_temp_edges:
        grid[e[0]][e[1]] = 10
    for e in hi_temp_edges:
        if e in lo_temp_edges:
            grid[e[0]][e[1]] = 45 # corner joining high- and lo-temp edges
        else:
            grid[e[0]][e[1]] = 80

angle    = 72.0/180*pi          # 72 degrees
# use complex plane to find the vertices
r        = exp(angle*1j)
# pentagon vertices in the complex plane, with first vertex duplicated
# at the end of the list
corners  = array([r**(i+0.25) for i in range(6)])
# translate pentagon into first quadrant
corners -= complex(min(real(corners)), min(imag(corners)))
corners *= 50                   # grid spacing (bigger means finer spacing)
center   = sum(corners[0:5])/5  # center of pentagon

# use Bresenham's algorithm to find the lattice points on the edges
lo_temp_edges = []
hi_temp_edges = []
for i in range(4):
    lo_temp_edges += line(complex2pair(corners[i]), complex2pair(corners[i+1]))
hi_temp_edges = line(complex2pair(corners[4]), complex2pair(corners[5]))

# figure out the grid dimensions
max_x = max([r[0] for r in lo_temp_edges+hi_temp_edges])
max_y = max([r[1] for r in lo_temp_edges+hi_temp_edges])
grid  = zeros((max_x+1,max_y+1))

dirs = [(-1,0), (1,0), (0,1), (0,-1)]
while True:
    newgrid = zeros((max_x+1,max_y+1))
    set_edge_temps(grid)        # impose constraint
    for x in range(max_x+1):    # relax each location to avg of its neighbors
        for y in range(max_y+1):
```

```
            total = n = 0
            for d in dirs:      # use each neighbor that's within the grid
                try:
                    total += grid[x+d[0]][y+d[1]]
                    n += 1
                except: pass    # that neighbor was not inside the grid
            newgrid[x][y] = total/n # but save new value in a new grid
    grid, newgrid = newgrid, grid   # swap new and old grid
    # print temperature at the center of the pentagon
    print grid[intround(real(center))][intround(imag(center))]
```