ECE302 – HW3

Jonathan Lam

March 11, 2021

$$
\begin{aligned}
f_{X,Y}(x,y) &= 2 \\
f_X(x) &= 2x \\
f_Y(y) &= 2(1-y) \\
f_{X|Y}(x \mid y) &= \frac{1}{1-y} \\
f_{Y|X}(x \mid y) &= \frac{1}{x} \\
\mu_X &= E[X] = \frac{2}{3} \\
\mu_Y &= E[y] = \frac{1}{3} \\
E[X^2] &= \frac{1}{2} \\
E[Y^2] &= \frac{1}{6} \\
\sigma_X^2 &= E[X^2] - \mu_X^2 = \frac{1}{18} \\
\sigma_Y^2 &= E[Y^2] - \mu_Y^2 = \frac{1}{18} \\
E[XY] &= \frac{1}{4} \\
\sigma_{XY}^2 &= E[XY] - \mu_X \mu_Y = \frac{1}{36} \\
\rho_{XY} &= \frac{\sigma_{XY}^2}{\sigma_X \sigma_Y} = \frac{1}{2}
\end{aligned}
$$

1. $f_X(x) = 2x$, $f_{Y|X}\left(y \mid \frac{1}{3}\right) = \frac{1}{\frac{1}{3}} = 3$
2. No. $f_{X,Y}(x,y) \neq f_X(x) f_Y(y)$
3. $$\hat{Y}_{\text{MMSE}}(X) = E[Y \mid X] = \int_0^x y f(y \mid x)\, dy = \frac{X}{2}$$

4.

$$\text{MSE} = E[(\hat{Y}_{\text{MMSE}}(X) - Y)^2] = E\left[\left(\frac{X}{2} - Y\right)^2\right] = \frac{1}{24}$$

$$b = E[\hat{Y} - Y] = 0 \Rightarrow \text{ unbiased (as expected)}$$

5.

$$\begin{aligned}\hat{Y}_{\text{LMMSE}}(X) &= \mu_Y + \rho_{XY}\frac{\sigma_Y}{\sigma_X}(X - \mu_X) \\ &= \frac{X}{2} \quad (= \hat{Y}_{\text{MMSE}}(X)) \\ \text{MSE} &= \sigma_Y^2(1 - \rho_{XY}^2) = \frac{1}{24} \text{ (same as before)}\end{aligned}$$

ECE302 – HW6

Jonathan Lam

April 13, 2021

Suppose in a binary communication system messages $X = 0$ and $X = 1$ occur with priori probabilities 0.25 and 0.75, respectively. Suppose we make observation $R = X + N$, where N is a continuous-valued r.v. with a uniform p.d.f.:

$$p_N(n) = \begin{cases} \frac{2}{3}, & -\frac{3}{4} < n < \frac{3}{4} \\ 0, & \text{else} \end{cases}$$

1. Define and plot the conditional p.d.f.s for H_0 and H_1.

   Let H_0 be the hypothesis that $X = 0$, and H_1 be the hypothesis that $X = 1$.

   $$p_0(r) = p_R(r \mid H = H_0) = p_N(n)$$
   $$p_1(r) = p_R(r \mid H = H_1) = \begin{cases} \frac{2}{3}, & \frac{1}{4} < r < \frac{7}{4} \\ 0, & \text{else} \end{cases}$$

2. Find the likelihood ratio test for the minimum probability of error.

   The MAP rule chooses H_0 if:

   $$p_{R|H}(r \mid H_1)P_1 < p_{R|H}(r \mid H_0)P_0$$
   $$P(r \mid H_1)(0.75) < P(r \mid H_0)(0.25)$$

   Since $P(r \mid H_1)$ and $P(r \mid H_0)$ are both equal to the constant value $\frac{2}{3}$ in their respective domains, then H_1 will always be chosen in its domain, and H_0 will only be chosen otherwise. I.e., the decision boundary is $\eta = r = \frac{1}{4}$:

   $$\hat{H}(r) = \begin{cases} H_0, & -\frac{3}{4} < r < \frac{1}{4} \\ H_1, & \frac{1}{4} < r < \frac{7}{4} \end{cases}$$

3. Compute the corresponding probability of error.

$$\begin{aligned}
P_{err} &= P(\hat{H} = 1 \cap H_0) + P(\hat{H} = 0 \cap H_1) \\
&= \left[\int_{\frac{1}{4}}^{\frac{7}{4}} P(r \mid H_0)\, dr\right]\left(\frac{1}{4}\right) + \left[\int_{-\frac{3}{4}}^{\frac{1}{4}} P(r \mid H_1)\, dr\right]\left(\frac{3}{4}\right) \\
&= \left[\int_{\frac{1}{4}}^{\frac{3}{4}} \frac{2}{3}\, dr\right]\left(\frac{1}{4}\right) + \left[\int_{\frac{1}{4}}^{\frac{1}{4}} \frac{2}{3}\, dr\right]\left(\frac{3}{4}\right) \\
&= \left[\frac{1}{2} \cdot \frac{2}{3}\right]\left(\frac{1}{4}\right) + 0\left(\frac{3}{4}\right) \\
&= \frac{1}{12}
\end{aligned}$$

4. Plot the receiver operating curve.

   The ROC is the locus of points $(P_F(\eta), P_D(\eta))$, where $P_F(\eta)$ is the conditionally probability of false alarm (Type I error), and $P_D(\eta)$ is the conditional probability of detection; both require the decision boundary.

$$\begin{aligned}
P_F(\eta) &= P(\hat{H} = 1 \mid H_0) = \int_{\eta}^{\infty} P(r \mid H_0)\, dr \\
P_D(\eta) &= P(\hat{H} = 1 \mid H_1) = \int_{\eta}^{\infty} P(r \mid H_1)\, dr
\end{aligned}$$

   The decision points of interest are $\eta = \{-\frac{3}{4}, \frac{1}{4}, \frac{3}{4}, \frac{7}{4}\}$. Since the distributions are uniform, we can expect that there will be a linear interpolation between these points on the ROC. Using the formulas above, we get the curve defined by:

$$(1,1), \left(\frac{1}{3}, 1\right), \left(0, \frac{2}{3}\right), (0,0)$$

ECE302 – Project 1 Analytical Results

Steven Lee & Jonathan Lam

February 3, 2021

Accompanying code can be found on GitHub.

Let X_i be the random variable denoting the roll of a single die; let $Y_j = X_1 + X_2 + X_3$ be the random variable denoting the roll of three die to generate an ability score; let $Z = \max(Y_1, Y_2, Y_3)$ be the random variable denoting the maximum of three trials to generate an ability score (using the "fun" method). All X_i are independent uniform IID; thus all Y_j are independent uniform IID, and all Z_k are independent uniform IID.

1. (a) Three Bernoulli trials

$$P(Y = 18) = P(X_1 = 6, X_2 = 6, X_3 = 6) = \frac{1}{6}\frac{1}{6}\frac{1}{6} = \frac{1}{6^3}$$

   (b) Complement of a binomial distribution

$$P(Z = 18) = 1 - P((Y_1 \neq 18) \wedge (Y_2 \neq 18) \wedge (Y_3 \neq 18)) = 1 - \left(1 - \frac{1}{6^3}\right)^3$$

   (c) Each trait is a Bernoulli trial

$$P(Z_i = 18,\ 1 \leq i \leq 6) = (P(Z = 18))^6$$

   (d) Get the set where all sums are ≤ 9, but at least one is equal to 9

$$\begin{aligned} P(Z = 9) &= P(\text{all sums } \leq 9) - P((\text{all sums } \leq 9) \wedge (\text{no sums } = 9)) \\ &= P((Y_1 \leq 9) \wedge (Y_2 \leq 9) \wedge (Y_3 \leq 9)) \\ &\quad - P((Y_1 \leq 8) \wedge (Y_2 \leq 8) \wedge (Y_3 \leq 8)) \\ &= \left(\frac{81}{216}\right)^3 - \left(\frac{56}{216}\right)^3 \end{aligned}$$

$$P(Z_i = 9,\ 1 \leq i \leq 6) = (P(Z = 9))^6$$

2. Let X_i denote the random variable representing the hitpoints (hp) of a goblin, and Y_j denote the random variable representing the damage (dmg) of a fireball shot. Note that $X_i \cup Y_j$ are mutually independent.

   (a) Normal expected value

$$\begin{aligned} E[X] &= \sum_x xP(X = x) \\ &= 1(0.25) + 2(0.25) + 3(0.25) + 4(0.25) \\ &= 2.5 \\ E[Y] &= \sum_y yP(Y = y) \\ &= 2(0.25) + 3(0.5) + 4(0.25) \\ &= 3 \\ E[Y > 3] &= P(Y = 4) = 0.25 \end{aligned}$$

   (b) We can enumerate the pmf by inspection

$$\begin{aligned} P(X = 1) = P(X = 2) = P(X = 3) = P(X = 4) = 0.25 \\ P(Y = 2) = P(Y = 4) = 0.25 \\ P(Y = 3) = 0.5 \end{aligned}$$

   (c) We break up this question using a partition of Y.

$$\begin{aligned} P(\text{slay all 6}) &= P(Y \geq X_i \ \forall X_i) \\ &= P((Y \geq X \ \forall X_i | Y = 2) \vee (Y \geq X \ \forall X_i | Y = 3) \\ &\quad \vee (Y \geq X \ \forall X_i | Y = 4)) \\ &= \left(\frac{1}{2}\right)^6 (0.25) + \left(\frac{3}{4}\right)^6 (0.5) + 1^6(0.25) \end{aligned}$$

(d) We can break down the event in question into a partition of three events (note that it is not possible that the surviving troll has ≤ 2 hp or that the firebolt did 4 dmg):

i. hp of surviving troll $= 4$, dmg $= 3$, all other trolls have hp ≤ 3
ii. hp of surviving troll $= 4$, dmg $= 2$, all other trolls have hp ≤ 2
iii. hp of surviving troll $= 3$, dmg $= 2$, all other trolls have hp ≤ 2

The probabilities of these events are easy to calculate.

Using Bayes' rule, we can calculate the posterior pmf of X given that five trolls didn't survive. Let W denote the event that the other five trolls died, and $W = (\text{i}) \cup (\text{ii}) \cup (\text{iii}) \Rightarrow P(W) = P((\text{i})) + P((\text{ii})) + P((\text{iii}))$ (union becomes addition since the events are disjoint). Then:

$$P(X = x|W) = \frac{P((X = x) \wedge W)}{P(W)}$$

This in turn can be used to calculate the expected hp of the surviving troll:

$$P((\text{iii})) = \left(\frac{1}{4}\right)\left(\frac{1}{4}\right)\left(\frac{1}{2}\right)^5$$

$$P((\text{ii})) = \left(\frac{1}{4}\right)\left(\frac{1}{2}\right)\left(\frac{3}{4}\right)^5$$

$$P((\text{i})) = \left(\frac{1}{4}\right)\left(\frac{1}{4}\right)\left(\frac{1}{2}\right)^5$$

$$P(X = 3|W) = \frac{P((\text{iii}))}{P(W)}$$

$$P(X = 4|W) = \frac{P((\text{i}) \cup (\text{ii}))}{P(W)}$$

$$E[X|W] = \frac{1}{P(W)}\left[(3)P(X = 3|W) + (4)P(X = 4|W)\right]$$

(e) Let Z_i denote the random variable denoting a roll of the 20-sided die (to decide whether Shedjam can hit Keene or not), W_j denote a roll of the 6-sided die (the Sword of Tuition's damage), and V_k denote a roll of the 4-sided die (the Hammer of Tenure Denial's damage).

$$\begin{aligned} E[\text{dmg}] &= E[\text{dmg}_{SoT} + \text{dmg}_{HTD}] \\ &= P(\text{hit}_{SoT})E[\text{dmg}_{SoT}|\text{hit}_{SoT}] + P(\text{hit}_{HTD})E[\text{dmg}_{HTD}|\text{hit}_{HTD}] \\ &= \left(\frac{10}{20}\right)(3.5 + 3.5) + \left(\frac{10}{20}\frac{10}{20}\right)(2.5) \end{aligned}$$

Stochastics Quiz #1
Name:
10 points total, 5 points each

Question 1: Suppose you have $63 and go to the casino, and play a game of roulette that only has 2 possible outcomes, red and green, such that P[red] = P[green] = ½. You employ the following strategy. First, you bet 1$ on red or green, if you guess correctly, you win $1, and leave the casino with $64. If you lose, you double down, and bet $2, in which case, if you guess correctly, you win $4. If you win, you take your winnings and go home. You keep doubling down until you either run out of money, or win once and then you stop.

a) What is the sample space of possible outcomes? (i.e. if you do this experiment, what possible winnings could you have?)
b) Write a PMF for the possible win amounts.
c) Find the average win amount?
d) Is this a good strategy? Should you play this game every day?

Question 2: *U* is a uniform random variable from (0,1). Let

$$X = -\ln(1-U)$$

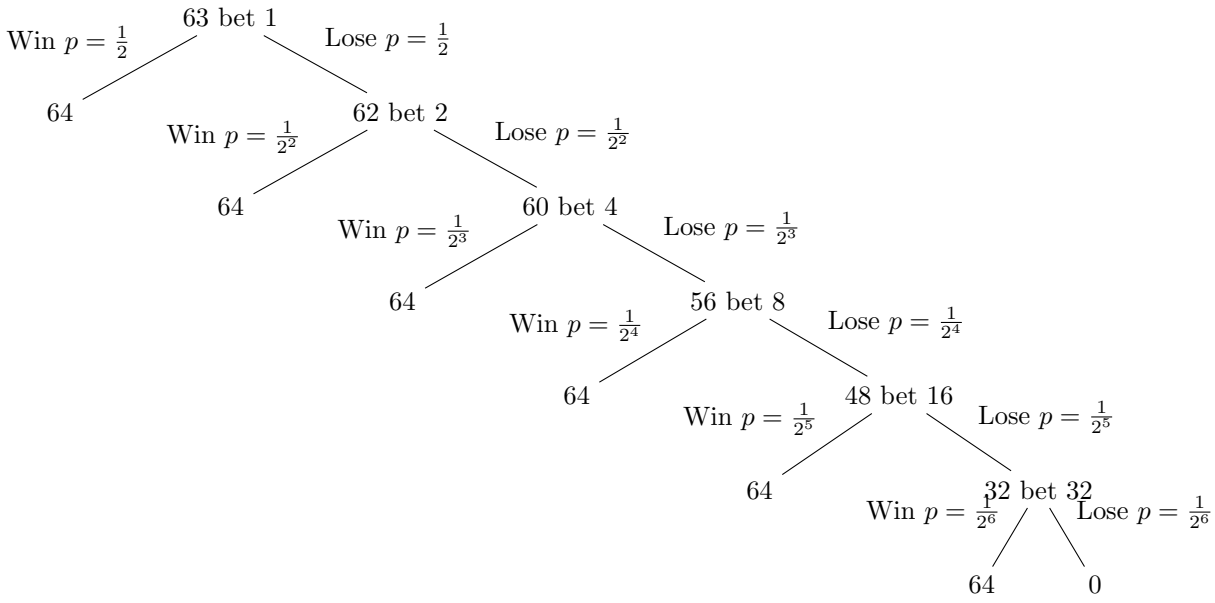
Find the cdf and pdf of *X*

ECE302 – Quiz 1

Jonathan Lam

February 11, 2021

1. Your possible winnings look like:



   (a) There are two possible outcomes: win one dollar (leave with 64 dollars) or lose all 63 dollars. (I.e., it is essentially a Bernoulli trial.)

   (b) Looking at the leaf nodes, we can sum the probabilities of winning one dollar (all disjoint events) to get the total probability of winning one dollar, and there is only one event where we lose all 63 dollars. Let X denote the Bernoulli R.V. indicating our winnings.

$$\begin{aligned} P[X = 1] &= \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} + \frac{1}{2^6} && = \frac{63}{64} \\ P[X = -63] &= \frac{1}{2^6} && = \frac{1}{64} \end{aligned}$$

(c) Expected value of X:

$$E[X] = (1)\frac{63}{64} + (-63)\frac{1}{64} = \frac{63}{64} - \frac{63}{64} = 0$$

(d) Since the expected value of the winnings is breaking even, it is up to you to push your luck if you want to win money. (Personally I wouldn't bother with something that isn't expected to earn anything in the long run. There's a good chance that you'd hit the -63 dollars at some point and make it so that you can't even continue to play anymore unless you're willing to run some debt.)

2. If U is a uniform R.V. from $(0, 1)$, then its cdf is:

$$F_U(u) = P[U \leq u] = \begin{cases} 0 & u < 0 \\ u & 0 \leq u < 1 \\ 1 & u \geq 1 \end{cases}$$

The cdf of X is then:

$$\begin{aligned} F_X(x) &= P[X \leq x] \\ &= P[-\ln(1-U) \leq x] \\ &= P[\ln(1-U) \geq -x] \\ &= P[1-U \geq e^{-x}] \\ &= P[U-1 \leq -e^{-x}] \\ &= P[U \leq 1-e^{-x}] \\ &= F_U(1-e^{-x}) \\ &= \begin{cases} 0 & x \leq 0 \\ 1-e^{-x} & x > 0 \end{cases} \end{aligned}$$

(Checking bounds on X: X is only valid when $U < 1$, i.e., for $1-e^{-x} < 1$, which is always true for all values of x.)

The pdf of X is the derivative of its cdf (use chain rule):

$$f_X(x) = \frac{dF_X}{dx} = \frac{dF_X}{d(1-e^{-x})}\frac{d(1-e^{-x})}{dx} = f_U(1-e^{-x})e^{-x}$$

Since $0 \leq 1-e^{-x} < 1$ for positive x and $1-e^{-x} < 0$ for negative x, and the pdf of U is:

$$f_U(u) = \begin{cases} 1 & 0 \leq u < 1 \\ 0 & \text{else} \end{cases}$$

then the pdf of X can be simplified to:

$$f_X(x) = \begin{cases} e^{-x} & x \geq 0 \\ 0 & \text{else} \end{cases}$$

ECE302 – Quiz 2

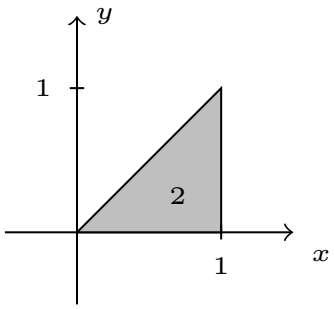
Jonathan Lam

February 25, 2021

1. Consider random variables X and Y that have the following joint pdf:

$$f_{X,Y}(x,y) = \begin{cases} 2 & 0 \le y \le x < 1 \\ 0 & \text{else} \end{cases}$$

Drawing of support of X, Y:



In the following answers, the intervals for which the pdfs are well-defined are shown in parentheses following their equations; outside those intervals the pdfs have value 0.

(a) Find the marginal pdfs $f_X(x)$ and $f_Y(y)$.

$$\begin{aligned} f_X(x) &= \int_{-\infty}^{\infty} f_{X,Y}(x,y')\,dy' \\ &= \int_0^x 2\,dy' \\ &= 2y'\Big|_0^x = 2x \ (0 \le x < 1) \end{aligned}$$

$$
\begin{aligned}
f_Y(y) &= \int_{-\infty}^{\infty} f_{X,Y}(x', y)\, dx' \\
&= \int_{y}^{1} f_{X,Y}(x', y)\, dx' \\
&= \int_{y}^{1} 2\, dx' \\
&= 2x' \Big|_{y}^{1} = 2(1-y) \ (0 \le y < 1)
\end{aligned}
$$

(b) Find the conditional pdfs $f_{Y|X}(y \mid x)$ and $f_{X|Y}(x \mid y)$.

$$
\begin{aligned}
f_{Y|X}(y \mid x) &= \frac{f_{X,Y}(x,y)}{f_x(x)} \\
&= \frac{2}{2x} = \frac{1}{x} \ (0 < y \le x \le 1) \\
f_{X|Y}(x \mid y) &= \frac{f_{X,Y}(x,y)}{f_y(y)} \\
&= \frac{2}{2(1-y)} = \frac{1}{1-y} \ (0 \le y \le x < 1)
\end{aligned}
$$

(c) Find the expected values of the conditional pdfs $E[X \mid Y]$ and $E[Y \mid X]$.

$$
\begin{aligned}
E[X \mid Y] &= \int_{-\infty}^{\infty} x f_{X|Y}(x \mid y)\, dx \\
&= \int_{y}^{1} x \frac{1}{1-y}\, dx \\
&= \frac{1}{1-y} \ \frac{x^2}{2} \Big|_{x=y}^{1} = \frac{1^2 - y^2}{2(1-y)} \\
&= \frac{y+1}{2} \ (0 < y \le 1) \\
E[Y \mid X] &= \int_{-\infty}^{\infty} y f_{Y|X}(y \mid x)\, dy \\
&= \int_{0}^{x} y \frac{1}{x}\, dy \\
&= \frac{1}{x} \ \frac{y^2}{2} \Big|_{y=0}^{x} = \frac{x}{2} \ (0 < x \le 1)
\end{aligned}
$$

Stochastics
Quiz 3, Spring 2021
Name:

Suppose *X* and *Y* are related by the following joint pdf:

$$p_{X,Y}(x,y) = \begin{cases} 10x & 0 \le x \le y^2, 0 \le y \le 1 \\ 0 & otherwise \end{cases}$$

a) Find the MMSE estimate of *X* based on *Y*.
b) Find the corresponding value of the mean squared error of this estimate.
c) Find the linear MMSE estimate of *X* based on *Y*.
d) Find the corresponding value of the mean squared the linear estimate.

ECE302

QUIZ 3

Jonathan Lam

a) $\hat{X}_{MMSE}(Y) = \frac{2}{3}Y^2$

b) $MSE_{\hat{X}_{MMSE}} = \frac{5}{162}$

c) $\hat{X}_{LMMSE}(Y) = -\frac{5}{14} + Y$

d) $MSE_{\hat{X}_{LMMSE}} = \frac{55}{1764}$

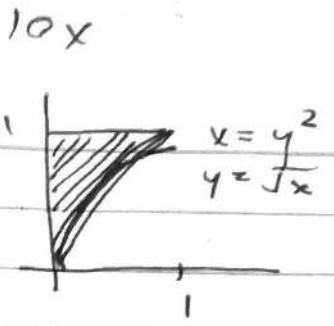
Note that $MSE_{\hat{X}_{MMSE}} \approx 0.0309$

is only slightly smaller than $MSE_{\hat{X}_{LMMSE}} \approx 0.0311$

which shows that linear estimate is quite good

$$\hat{y}_{MMSE}(x) = E[y|x]$$

$$= \int_0^1 y\, f(y|x)\, dy$$



$$f_{x,y}(x,y) = \begin{cases} 10x, & 0 \le x \le y^2,\ 0 \le y \le 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$f_x(x) = \int_0^1 f(x,y)\, dy$$

$$= 10xy \Big|_{y=0}^{1} = 10x$$

$$f_x(x) = \int_{\sqrt{x}}^1 10x\, dx$$

$$= 10xy \Big|_{\sqrt{x}}^{1} = 10x(1-\sqrt{x})$$

$$f_y(y) = \int_0^{y^2} 10x\, dx$$

$$= 5 \cdot \frac{10x^2}{2} \Big|_0^{y^2} = 5y^4.$$

$$f(y|x) = \frac{f(x,y)}{f(x)} = \frac{10x}{10x(1-\sqrt{x})} = \frac{1}{1-\sqrt{x}}$$

$$E[y|x] = \int_{\sqrt{x}}^1 y\, f(y|x)\, dy$$

$$= \int_{\sqrt{x}}^1 y \frac{1}{1-\sqrt{x}}\, dy$$

$$= \frac{1}{1-\sqrt{x}} \frac{y^2}{2}\Big|_{\sqrt{x}}^{1}$$

$$= \frac{1}{1-\sqrt{x}} \left(\frac{1}{2} - \frac{x}{2} \right)$$

$$= \frac{1-x}{2(1-\sqrt{x})} = \frac{1}{2}\left(\frac{1^2 - \sqrt{x}^2}{1-\sqrt{x}} \right)$$

$$= \frac{1}{2} \frac{(1-\sqrt{x})(1+\sqrt{x})}{1-\sqrt{x}}$$

$$= \frac{1+\sqrt{x}}{2}.$$

$$E\left[\left(\hat{Y}_{MMSE}(X) - Y\right)^2\right] = E\left[\left(\frac{1+\sqrt{X}}{2} - Y\right)^2\right]$$

$$= E\left[\left(\frac{1+\sqrt{X}}{2}\right)^2\right]$$

$$f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} = \frac{10x}{5y^4} = \frac{2x}{y^4}$$

$$E[X|Y] = \int_0^{y^2} x\, f(x|y)\, dx$$

$$= \int_0^{y^2} x \left(\frac{2x}{y^4}\right) dx = \frac{1}{y^4} \int_0^{y^2} 2x^2\, dx$$

$$= \frac{1}{y^4} \frac{2x^3}{3}\Big|_0^{y^2}$$

$$= \frac{1}{y^4} \frac{2y^6}{3} = \frac{2}{3} y^2.$$

$\hat{X}_{mmse}(Y) = E[X|Y] = \frac{2}{3}Y^2$

$E[(\hat{X}_{mmse}(Y) - X)^2]$

$= E[(\frac{2}{3}Y^2 - X)^2]$

$= E[\frac{4}{9}Y^4 - 2(\frac{2}{3}Y^2X) + X^2]$

$= \frac{4}{9}E[Y^4] - \frac{4}{3}E[Y^2X] + E[X^2]$

$= \frac{4}{9}\int_0^1 y^4 f_Y(y)\,dy - \frac{4}{3}\int_0^1\int_0^{y^2} y^2 x f_{X,Y}(x,y)\,dx\,dy + \int_0^1 x^2 f_X(x)\,dx$

$= \frac{4}{9}\int_0^1 y^4(5y^4)\,dy - \frac{4}{3}\int_0^1\int_0^{y^2} y^2 x(10x)\,dx\,dy + \int_0^1 x^2(10x)(1-\sqrt{x})\,dx$

$= \int 5y^8\,dy$

$\Rightarrow \frac{5y^9}{9}\Big|_0^1$

$= \frac{5}{9}$

$\int_0^1 10x^3 - 10x^{3.5}\,dx$

$= \frac{10x^4}{4} - \frac{10x^{4.5}}{4.5}\Big|_0^1$

$= \frac{10}{4} - \frac{10}{4.5}$

$= \frac{5}{2} - \frac{20}{9}$

$= \frac{45}{18} - \frac{40}{18} = \frac{5}{18}$

$\frac{4}{9}\cdot\frac{5}{9} = \frac{20}{81}$

$\frac{7}{3}$ $10\int_0^1 \int_0^{y^2} y^2 x^2 \, dx \, dy$

$= 10\frac{4}{3}\int_0^1 y^2 \int_0^{y^2} x^2 \, dx \, dy$

$= 10\frac{4}{3}\int_0^1 y^2 \left.\frac{x^3}{3}\right|_0^{y^2} dy$

$= 10\frac{4}{3}\int_0^1 y^2 \frac{y^6}{3} \, dy$

$= \frac{40}{9}\int_0^1 y^8 \, dy$

$= \frac{40}{9} \cdot \left.\frac{y^9}{9}\right|_0^1$

$= \frac{40}{81}$.

$= \frac{20}{81} - \frac{40}{81} + \frac{5}{18}$

$= \frac{-20}{81} + \frac{5}{18} = \frac{-40}{162} + \frac{45}{162}$

$= \frac{5}{162}$.

LMMSE estimate:

$Y_{LMMSE}(x) = a_0 + a_1 X$

$a_0 = E[Y] - a_1 E[X]$

$a_1 = \frac{\sigma_{XY}}{\sigma_X^2} = \frac{\rho_{XY}\sigma_X\sigma_Y}{\sigma_X^2} = \rho_{XY}\frac{\sigma_Y}{\sigma_X}$

we are estimating X based on Y, so have to flip this.

Need to find:

σ^2_x, σ^2_y, σ_{xy}, μ_x, μ_y.

$$\mu_x = \int_0^1 x f_x(x)\,dx$$

$$= \int_0^1 x\left(10x\left(1-\sqrt{x}\right)\right)dx$$

$$= \int_0^1 10x^2 - 10x^{2.5}\,dx$$

$$= \frac{10x^3}{3} - \frac{10x^{3.5}}{3.5}\Big|_0^1$$

$$= \frac{10}{3} - \frac{10}{3.5} = \frac{10}{3} - \frac{20}{7}$$

$$= \frac{70}{21} - \frac{60}{21} = \frac{10}{21}.$$

$$\mu_y = \int_0^1 y f_y(y)\,dy$$

$$= \int_0^1 y(5y^4)\,dy$$

$$= \int_0^1 5y^5\,dy = \frac{5}{6}.$$

$E[x^2] = \frac{5}{18}.$

$$E[y^2] = \int_0^1 y^2 f_y(y)\,dy$$

$$= \int_0^1 y^2(5y^4)\,dy$$

$$= \int_0^1 5y^6\,dy = \frac{5}{7}$$

$\sigma^2_x = E[x^2] - \mu_x^2$

$= \frac{5}{18} - \left(\frac{10}{21}\right)^2$

$= \frac{5}{18} - \frac{100}{441}$

$2 \cdot 3^2$ $\quad$ $7^2 3^2$

$\frac{5.49}{882} - \frac{200}{882} = \frac{245 - 200}{882}$

$= \frac{45}{882} = \frac{5}{98}$

$\sigma^2_y = E[y^2] - \mu_y^2$

$= \frac{5}{7} - \left(\frac{5}{6}\right)^2 = \frac{5}{7} - \frac{25}{36}$

$\frac{180 - 175}{252} = \frac{5}{252}$

$\sigma_{xy} = E[XY] - \mu_x \mu_y$

$E[xy] = \int_0^1 \int_0^{y^2} xy(10x)\, dx\, dy$

$= 10 \int_0^1 y \int_0^{y^2} x^2\, dx\, dy$

$= 10 \int_0^1 y \left.\frac{x^3}{3}\right|_0^{y^2} dy$

$= \frac{10}{3} \int_0^1 y \cdot y^6\, dy$

$= \frac{10}{3} \left(\frac{y^8}{8}\right)\Big|_0^1 = \frac{10}{24} = \frac{5}{12}$.

$\sigma_{xy} = \frac{5}{12} - \frac{10}{21}\left(\frac{5}{63}\right)$

$= \frac{5}{12} - \frac{25}{63}$

$\frac{35}{84} - 100$

$\frac{105}{252} - \frac{100}{252} = \frac{5}{252}$

LMMSE

$a_1 = \frac{\sigma_{xy}}{\sigma^2_x} = \frac{\frac{5}{252}}{\frac{7}{18}}$

$= \frac{98}{252} = \frac{49}{126} = \frac{7}{18}$

$a_0 = E[y] - a_1 E[x]$

$= \frac{5}{6} - \frac{7}{18}\left(\frac{10}{21}\right)$

$= \frac{5}{6} - \frac{5}{27}$

$\frac{45}{54} - \frac{10}{54} = \frac{35}{54}$

$$\hat{X}_{LMMSE}(Y) = a_0 + a_1 Y$$

$$a_1 = \frac{\sigma_{XY}}{\sigma^2_Y} \qquad a_0 = \mu_X - a_1 \mu_Y$$

$$a_1 = \frac{\frac{5}{252}}{\frac{5}{252}} = 1.$$

$$a_0 = \frac{10}{21} - (1)\frac{5}{6}$$

$$= \frac{20}{42} - \frac{35}{42} = -\frac{15}{42}$$

$$= \frac{-5}{14}.$$

$$MSE_{LMMSE} = E\left[\left(\hat{X}_{LMMSE}(Y) - X\right)^2\right]$$

$$= \sigma^2_X (1 - \rho^2_{XY})$$

$$\rho = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \frac{\frac{5}{252}}{\sqrt{\frac{5}{98}}\sqrt{\frac{5}{252}}} =$$

$$\frac{\sqrt{252-98}}{252} = \sqrt{\frac{98}{252}} = \sqrt{\frac{7}{18}}$$

$$MSE_{LMMSE} = \frac{5}{98}\left(1 - \left(\sqrt{\frac{7}{18}}\right)^2\right)$$

$$= \frac{5}{98}\left(1 - \frac{7}{18}\right) = \frac{5}{98} \cdot \frac{11}{18}$$

$$= \frac{55}{1764}$$

$$\begin{array}{r} 6 \\ 98 \\ \times\ 18 \\ \hline 784 \\ 980 \\ \hline 1764 \end{array}$$

ECE302 – Quiz 4

Jonathan Lam

March 25, 2021

Let X be a Pareto R.V. with parameters α and x_m. Both α and x_m are positive, and x_m is the minimum possible value that X can take. The p.d.f. of a Pareto R.V. is as follows:

$$f_X(x) = \begin{cases} \frac{\alpha x_m^\alpha}{x^{\alpha+1}} & x \geq x_m \\ 0 & x < x_m \end{cases}$$

1. Find the M.L. estimate of α from a sequence of observations $\mathbf{X} = \{X_1, \ldots, X_n\}$, assuming x_m is known.

   Assume observations of $\mathbf{X}$ are I.I.D.

$$\begin{aligned} l(\mathbf{X} \mid \alpha, x_m) &= \prod_{j=1}^{n} f_X(X_j \mid \alpha) = \prod_{j=1}^{n} \frac{\alpha x_m^\alpha}{X_j^{\alpha+1}} \\ L(\mathbf{X} \mid \alpha, x_m) &= \ln l(\mathbf{X} \mid \alpha, x_m) \\ &= \sum_{j=1}^{n} \ln \alpha + \alpha \ln x_m - (\alpha + 1) \ln X_j \\ \frac{\partial L}{\partial \alpha} &= \sum_{j=1}^{n} \frac{1}{\alpha} + \ln x_m - \ln X_j \end{aligned}$$

   Maximize log-likelihood function by setting the derivative to zero.

$$\begin{aligned} \frac{\partial L}{\partial \alpha} &= 0 \\ n\left(\frac{1}{\alpha} + \ln x_m\right) &= \sum_{j=1}^{n} \ln X_j \\ \alpha &= \left(\frac{1}{n} \sum_{j=1}^{n} \ln X_j - \ln x_m\right)^{-1} \end{aligned}$$

2. Argue that if x_m is unknown, the M.L. estimate of this parameter is the minimum observed value of $\mathbf{X}$.

   x_m is bounded by above by $\min \mathbf{X}$ by definition of the Pareto R.V. We also see that the (log) likelihood function increases monotonically as a function of x_m. Thus the value of x_m that maximizes the likelihood function is the its maximum possible value, i.e., $\min \mathbf{X}$.

3. How does this change your answer to part (1)?

   If we know neither α nor x_m, then the M.L. estimate for α becomes

$$\alpha = \left(\frac{1}{n} \sum_{j=1}^{n} \ln X_j - \ln \min \mathbf{X} \right)^{-1}$$

ECE302 – Quiz 5

Jonathan Lam

April 15, 2021

A random variable R is observed, and it is known that under H_0:

$$f_0(r) = f_{R|H_0}(r \mid H_0) = \begin{cases} \frac{1}{2}, & -1 \leq r \leq 1 \\ 0, & \text{else} \end{cases}$$

and under H_1:

$$f_1(r) = f_{R|H_1}(r \mid H_1) = \frac{1}{2} e^{-|r|}$$

Furthermore, assume the priors $p_0 = \frac{1}{3}$ and $p_1 = \frac{2}{3}$.

1. Plot the class conditional p.d.f.'s on one axis. Be sure to label.

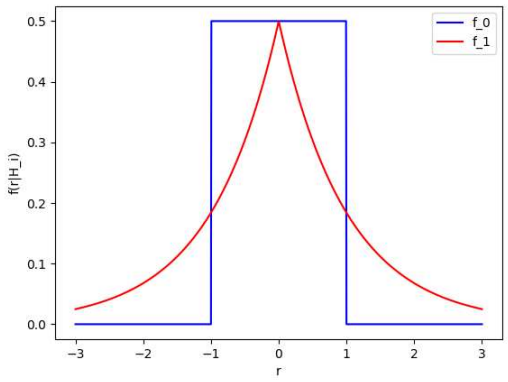


Figure 1: Plot of class conditional densities on the interval $-3 \leq r \leq 3$

2. Consider a decision rule: If $|r| > \gamma$ decide H_0 ($\hat{H} = 0$), else decide H_1 ($\hat{H} = 1$). Determine the probability of error for this rule if $\gamma = \frac{1}{2}$.

   Using the symmetry of the problem, we can focus on the positive half-line. The decision rule then becomes: if $0 \leq r \leq \frac{1}{2}$, then choose H_1; if $r > \frac{1}{2}$ choose H_0. Thus:

$$\begin{aligned} P_{err} &= P(\hat{H} = 1 \cap H_0) + P(\hat{H} = 0 \cap H_1) \\ &= \left[\int_0^{\frac{1}{2}} f(r \mid H_0)\, dr\right]\left(\frac{1}{3}\right) + \left[\int_{\frac{1}{2}}^{\infty} f(r \mid H_1)\, dr\right]\left(\frac{2}{3}\right) \\ &= \left[\int_0^{\frac{1}{2}} \frac{1}{2}\, dr\right]\left(\frac{1}{3}\right) + \left[\int_{\frac{1}{2}}^{\infty} \frac{1}{2} e^{-r}\, dr\right]\left(\frac{2}{3}\right) \\ &= \frac{1}{12} + \frac{1}{3} e^{-\frac{1}{2}} \approx 0.286 \end{aligned}$$

3. Determine the likelihood ratio test that minimizes the overall probability of error.

   Again, we focus only on the positive half-line because of the symmetry. The MAP rule states that the decision boundary should occur when:

$$f(r \mid H_1) p_1 = f(r \mid H_0) p_0$$

   In other words, choose the decision boundary η such that:

$$\begin{aligned} \left(\frac{1}{2} e^{-\eta}\right)\left(\frac{2}{3}\right) &= \left(\frac{1}{2}\right)\left(\frac{1}{3}\right) \\ e^{-\eta} &= \frac{1}{2} \\ \eta &= \ln 2 \end{aligned}$$

   Thus the MAP rule is: for $|r| > \eta$, choose H_0; otherwise choose H_1.

4. Now suppose p_0 and p_1 are no longer constrained, but cannot be either 0 or 1. Find which values of p_0 such that the decision rule that minimizes the probability of error always decides the same hypothesis, regardless of the observation.

   This situation occurs if the a posteriori probabilities (class conditional multiplied by the priors) don't ever overlap; i.e., the a posteriori probability of H_0 is always greater than H_1, or vice versa.

   It is not possible that the a posteriori probability of H_0 is always higher than H_1, because it is 0 for parts of the domain where H_1 is not. The opposite is true, if the a posteriori probability of H_1 is always greater than the a posteriori probability of H_0 for $|r| \leq 1$, and particularly at $|r| = 1$.

$$\left[\frac{1}{2} e^{-1}\right](1 - p_0) > \frac{1}{2} p_0 \Rightarrow p_0 < \frac{1}{e+1}$$

   The MAP decision rule always chooses H_1 if $p_0 < (e+1)^{-1}$.

SSSSS: Seventies-Style Sight and Sound System

Jonathan Lam (lam12@cooper.edu)
Steven Lee (lee70@cooper.edu)

The Cooper Union for the Advancement of Science and Art
ECE394 Electrical and Computer Engineering Projects II
Prof. Stuart Kirtman

May 14, 2021

Introduction

Figure: A Radio Shack advertisement for a color organ.

Block diagram

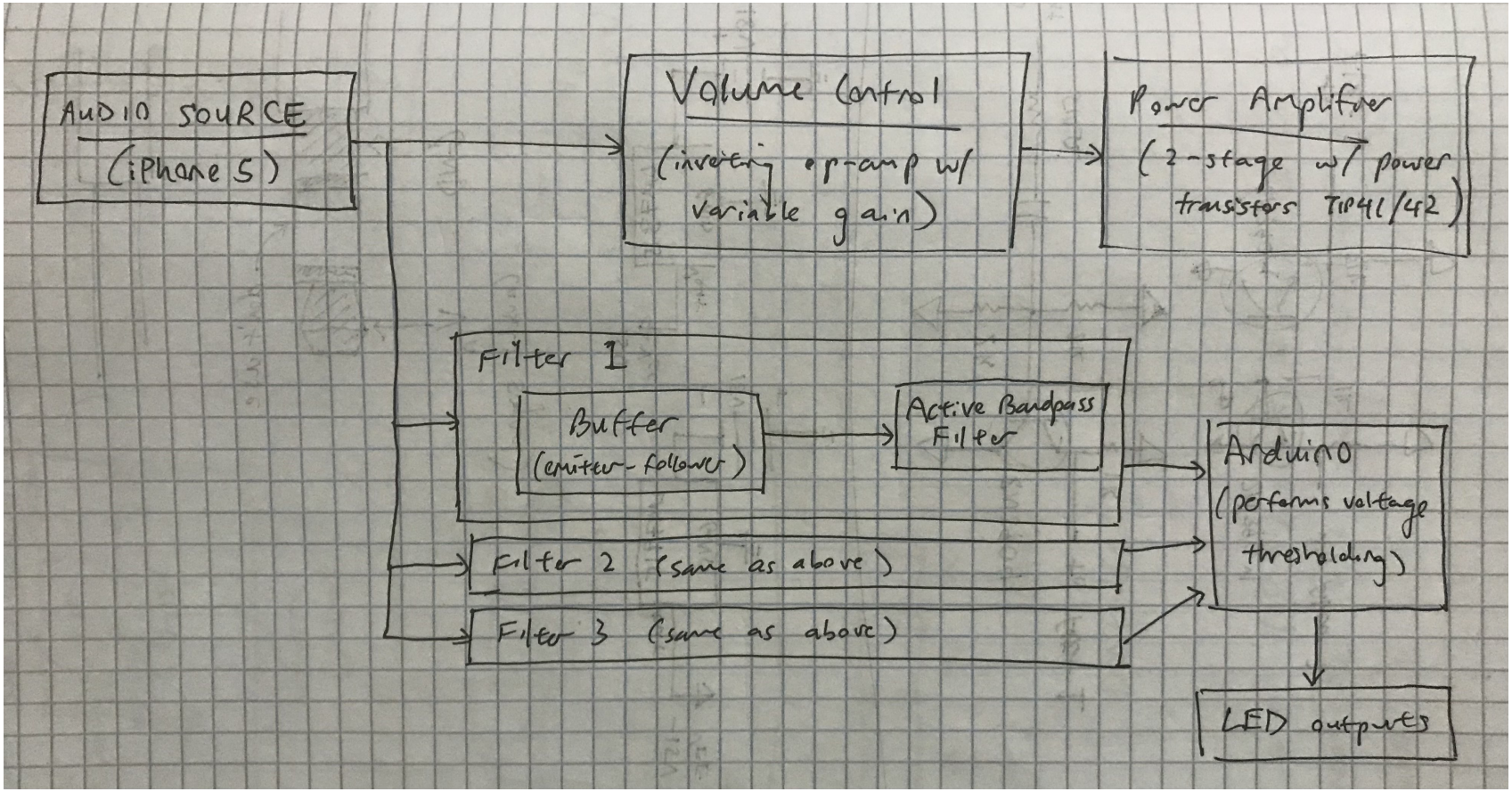


Figure: Block diagram of the overall color organ

Demonstrating the bandpass filter

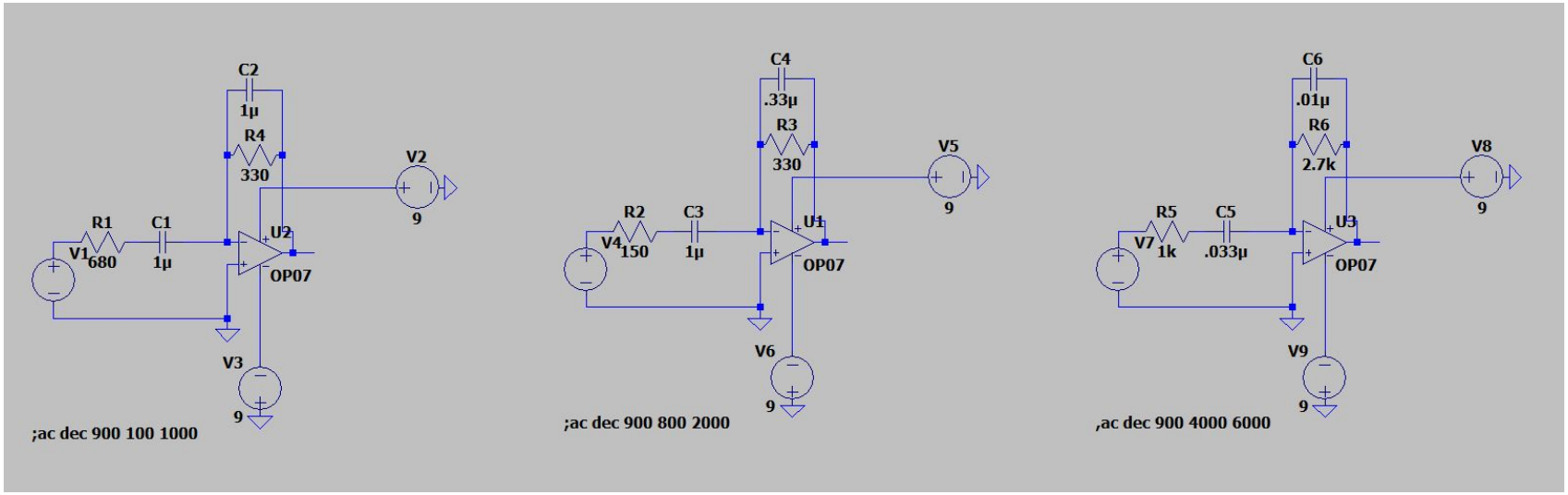


Figure: Schematic of the three filters we created.

Demonstrating the speaker circuit

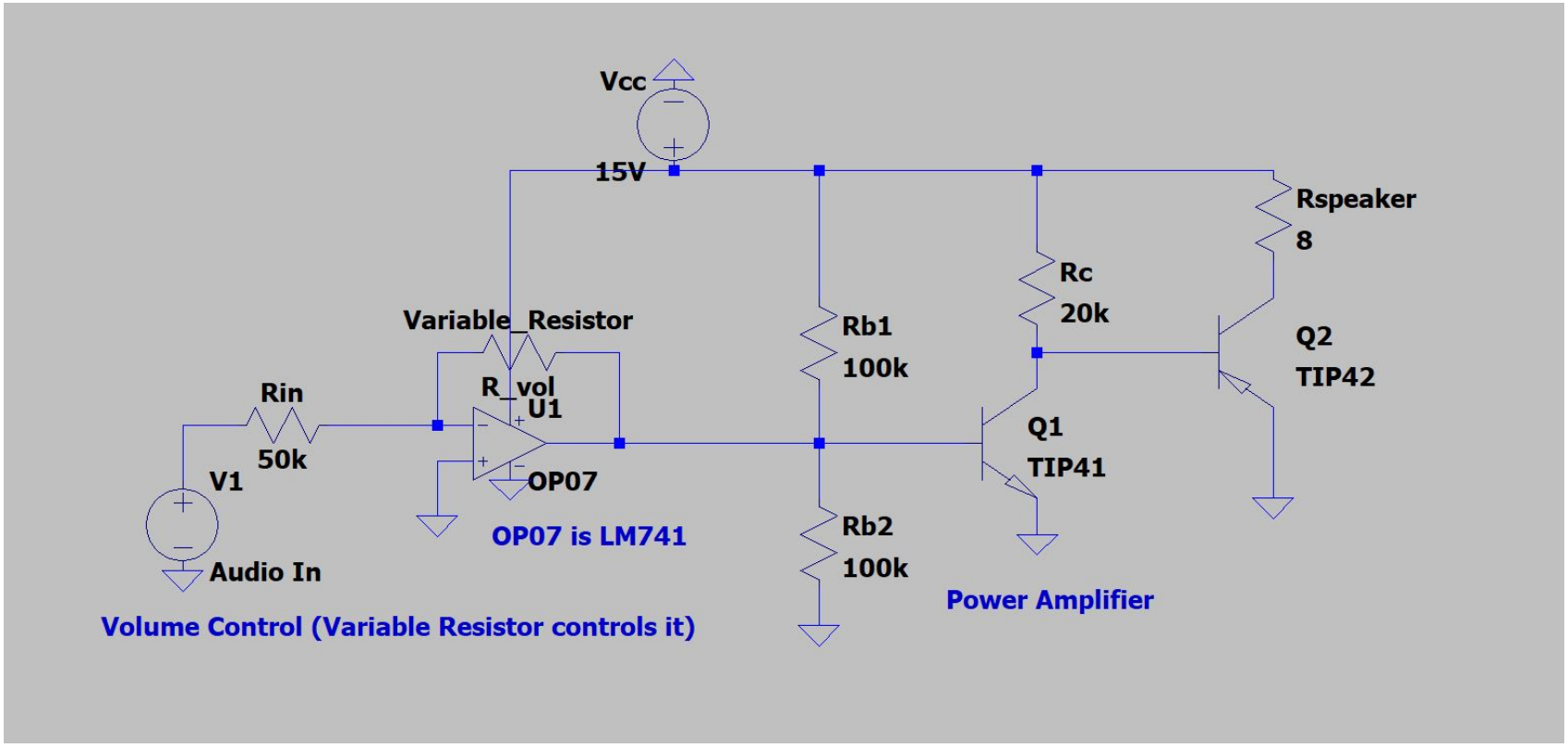


Figure: Left: a regular inverting amplifier for volume control. Right: a Sziklai pair using power transistors to drive the 8 Ω speaker.

Difficulties

- Generating enough power for speaker (but not too much)
- Choosing capacitor/resistor values for filters with given parts
- Preventing major components from affecting each other

Conclusions and next steps

- Was harder than we expected (when we put the pieces together)
- Figure out why third filter doesn't work
- Make this work better for all amplitudes
- Also implement thresholding in analog (comparator)

Thanks for watching

SSSSS: Seventies-Style Sight and Sound System

Jonathan Lam (lam12@cooper.edu)
Steven Lee (lee70@cooper.edu)

The Cooper Union for the Advancement of Science and Art
ECE394 Electrical and Computer Engineering Projects II
Prof. Stuart Kirtman

May 14, 2021

1 Abstract

The color organ is an electronic speaker for music that has colored lights corresponding that light up based on the the frequency of an audio signal. The goal of this project is to implement a simple color organ with three frequency ranges: bass (225-450Hz), midrange (1000-1500Hz), and treble (4800-5500Hz).

2 Introduction

As described in the abstract, a color organ is a device that can detect different frequencies and light up different LEDs based on its input.

Color organ, a device that allows people to enjoy both music and make an relation with color was a creative invention during the 18th century for people to enjoy music[1]. Even though it was all manual at the time, it was a big hit and people loved associating the two together. Fast forward to the late 1960s, color organ showed up as a different form, with the ability to detect different frequency/amplitude to show different colors and making it a lot more easier to operate.

3 Block diagram

The block diagram is shown in Figure 1. The project comprised two major components: amplifying the signal enough to power an $8\,\Omega$ speaker, and filtering the signal into the respective bands. These will be described later in greater detail.

4 Schematic diagrams

4.1 Filter design

We have constructed three active bandpass filters for the design, where each one would be detecting a different frequency. The formula we use to determine this is:

$$\omega_c = \frac{1}{2\pi f c} \tag{1}$$

The schematics for each filter are shown in Figure 2.

[1] `https://en.wikipedia.org/wiki/Color_organ`

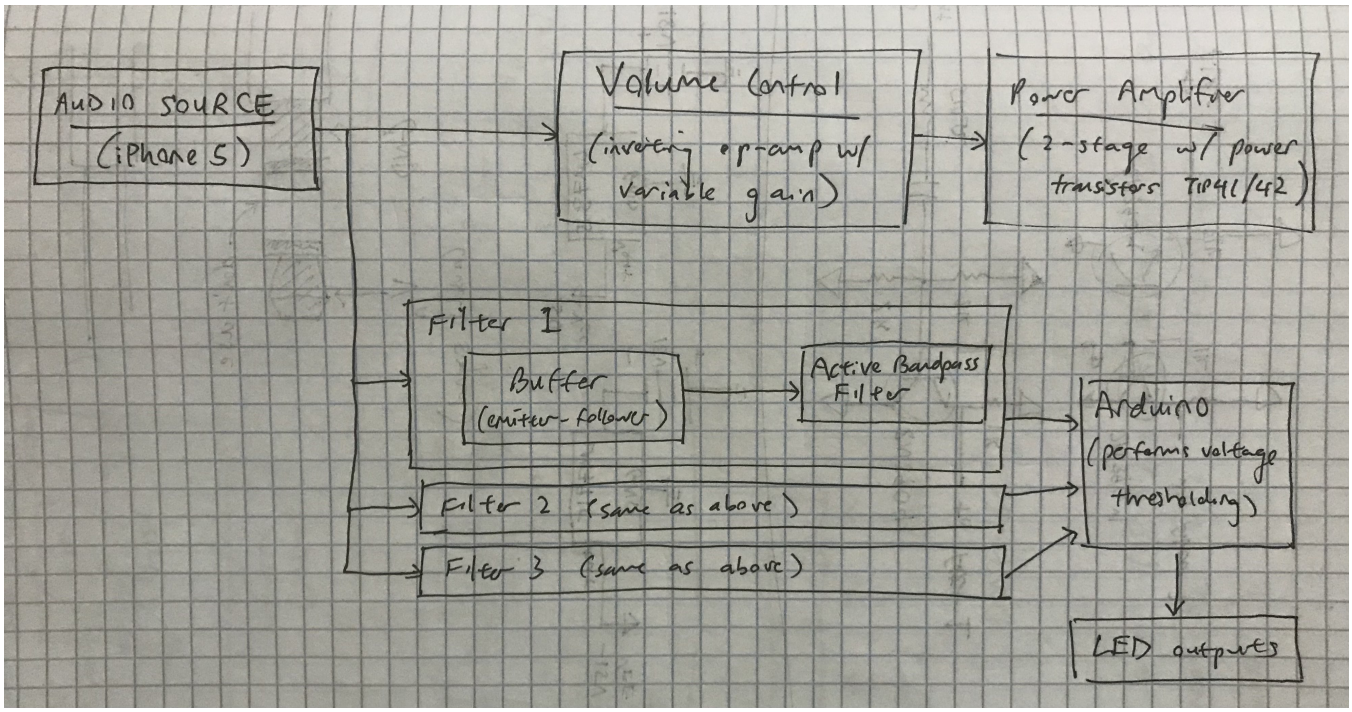


Figure 1: Block diagram of the overall color organ

4.2 Volume control and power amplification

For volume control, we use a simple inverting amplifier, where the feedback resistance is a 50 kΩ varistor. The gain for this is:

$$A_V = -\frac{R_{vol}}{R_{in}} \tag{2}$$

In the case of $R_{vol} = R_{in}$, this allows us to decrease the amplitude already. To allow for amplification, then R_{vol} should be greater than R_{in}.

To be able to drive the speaker, we need a power amplifier. Originally, we tried a single op-amp (LM741) amplifier to power the speaker, but this only resulted in noise at the output – clearly, the output of this ordinary op-amp is not powerful enough to drive the speaker. Similarly, a single BJT (2N3904) was not enough.

During the labs we discovered the Darlington pair topology, in which two NPN transistors are chained in a way such that their current gains are multiplied. This allows for very high current gains. In our kit we have Darlington pair power transistors TIP120 (NPN) and TIP125 (PNP). However, we were unable to find out how to implement this correctly in the circuit, due to a lack of examples online and a lack of experience with setting up Darlington pairs or power transistors.

We were able to use a topology similar to the Darlington pair called the Sziklai Pair (also known as the "Complementary Darlington") which uses a NPN and PNP. They offer several advantages over Darlington pairs, which are noted for example on the Wikipedia page[2]. Our reference for the schematic is from "DIY Amplifier using TIP41C and TIP42C transistors"[3]. This example used a 12V source and ground as its power rails (note that the VEE rail is not used for this subcircuit). We kept most of the same values except that we used the 15V from VCC, used a $20\,\mathrm{k}\Omega$ rather than a $18\,\mathrm{k}\Omega$ collector resistor, and made some minor adjustments to how the transistors were connected to VCC. The final schematic is shown in Figure 3.

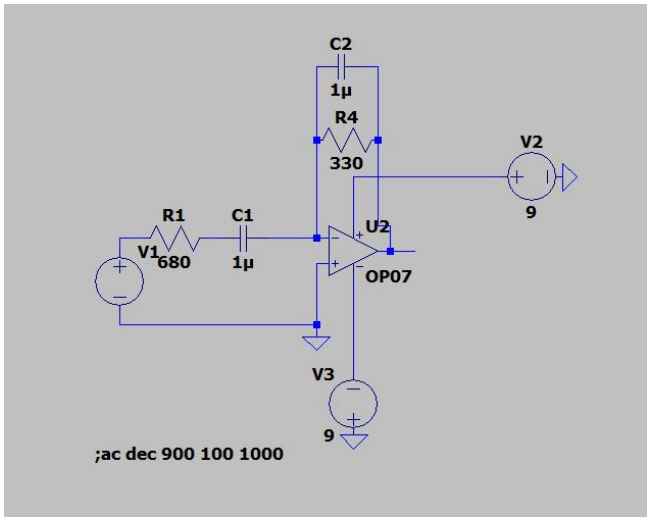
The result sounds fairly good, except that it causes the TIP42, the LM7815 (VCC voltage regulator), and the speaker to heat up. (At first, we did not know this, and it seems to have damaged one of our speakers so that the output sounds distorted and fainter.) As a result, the TIP42 and LM7815 were "mounted" onto a heatsink (via tape, as the solder didn't stick onto the copper heatsink without flux, which we didn't have on hand). This solved the heating issue for the chips, but the speaker still heated up very quickly, thus limiting us in how long we could run the color organ continuously. We couldn't figure out the exact computations to set up the power correctly, but this is something to be investigated for a practical amplifier circuit that would be run continuously for any reasonable period of time. More power efficient speakers (e.g., Class D speakers) are much more complicated and have more stages in order to achieve higher efficiency; our two-stage filter is probably loosely a Class A amplifier[4].

4.3 Power regulation

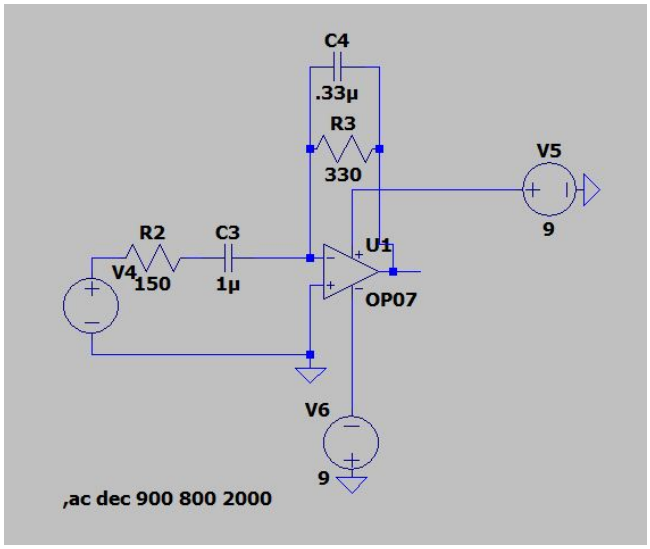
The VCC and VEE power rails were $\pm 15\,\mathrm{V}$. These each used two 9V batteries and the appropriate voltage regulator (LM7815 and LM7915, respectively). See Figure 4.

We had a few problems with the power rails. Firstly, there was a small periodic noise in the VEE rail, even when nothing but the voltage regulator was plugged into it. We smoothed this out with a decoupling capacitor connected between VEE and GND. Another issue was that the VCC rail dropped significantly when powering the speaker and power amplifier circuit – this was due to drained batteries and was fixed when the batteries were replaced.

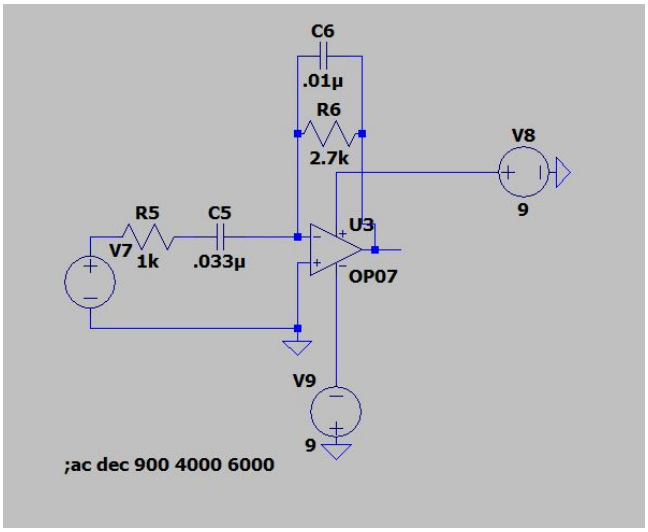
[2] `https://en.wikipedia.org/wiki/Sziklai_pair#Advantages`
[3] https://www.youtube.com/watch?v=jFf6VnkpF6Q
[4] `https://en.wikipedia.org/wiki/Power_amplifier_classes`



(a) Bass Filter: R1 = 680 Ω, C1 = 1 µF , R2 = 330 Ω, C2 = 1 µF



(b) Midrange Filter: R1 = 150 Ω, C1 = 1 µF , R2 = 330 Ω, C2 = 0.33 µF



(c) Treble Filter: R1 = 1 kΩ, C1 = 0.033 µF , R2 = 2.7 kΩ, C2 = 0.01 µF

Figure 2: Schematic diagrams for all active bandpass filters.

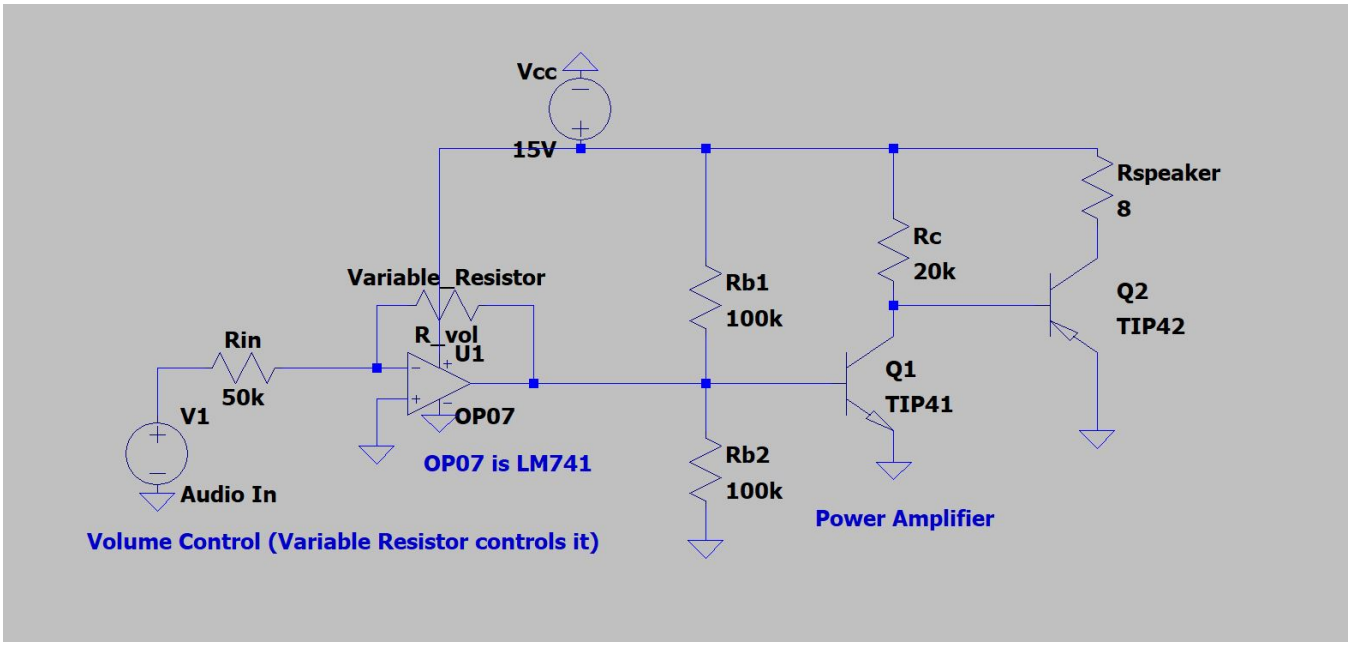


Figure 3: Left: a regular inverting amplifier for volume control. Right: a Sziklai pair using power transistors to drive the 8 Ω speaker.

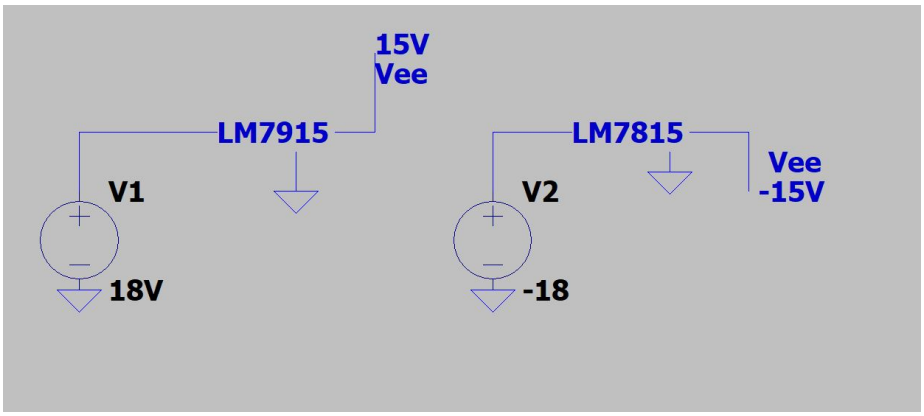


Figure 4: Voltage Regulator

5 Arduino script

For the purposes of this project, we were allowed to use an Arduino to do processing outside of the filtering (which had to be done in analog). We decided to use the Arduino to perform thresholding on the filtered signals to determine when to light the LEDs.

The Arduino script is very simple: for each filter output, choose a voltage threshold (empirically determined) above which the LED should turn on. The filter outputs are sampled uniformly in discrete time and used to determine whether the corresponding LED should turn on or off. Note that this simple sampling method doesn't detect extremas or zero-crossings, but is rather a random sampling of the possible voltages given the instantaneous signal amplitude. As a result, a hysteresis delay factor is used so that the LEDs don't immediately turn back off.

The Arduino is then connected to three digital outputs, which are connected to LEDs in series with a $220\,\Omega$ resistor. The schematics are not shown here because they are trivial.

An alternative (using analog components) would be to use a comparator (LM311) on the smoothed rectified output. We did not have enough time to try out this method, but we estimate that it will give better results than the Arduino's method due to the limited sampling rate of the Arduino, as compared with the high slew rate of the comparator and the continuous (smoothed) voltage.

```
int val = 0, iThreshold = 20, i = 0, j;

int inputs[] = {A0,A1,A2},
  outputs[] = {2,3,4},
  vThresholds[] = {25, 130, 60},
  lastIs[] = {0,0,0},
  ons[] = {0,0,0};

void setup() {
  for (j=0; j<3; ++j) {
    pinMode(inputs[j], INPUT);
    pinMode(outputs[j], OUTPUT);
  }
}

void loop() {
  ++i;

  for (j=0; j<3; ++j) {
    val = analogRead(inputs[j]);

    if (val > vThresholds[j]) {
      lastIs[j] = i;
      if (!ons[j]) {
        digitalWrite(outputs[j], 1);
        ons[j] = 1;
      }
    } else {
      if (ons[j] && i-lastIs[j]>iThreshold) {
        digitalWrite(outputs[j], 0);
        ons[j] = 0;
      }
    }
  }
}
```

Figure 5: Thresholding script for Arduino. `vThresholds` are found empirically and are (unfortunately) sensitive to environmental conditions. The counter variable `i` and the corresponding variable `lastIs` are used for hysteresis.

6 LTSpice simulations

See the simulations for each filter in Figures 6, 7, 8. An AC sweep is performed to simulate the frequency response of these filters.

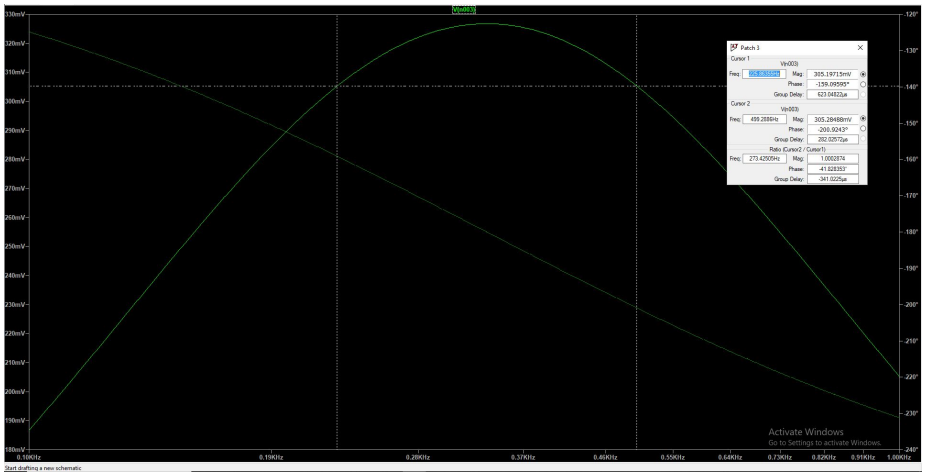


Figure 6: Simulation of Bass Filter

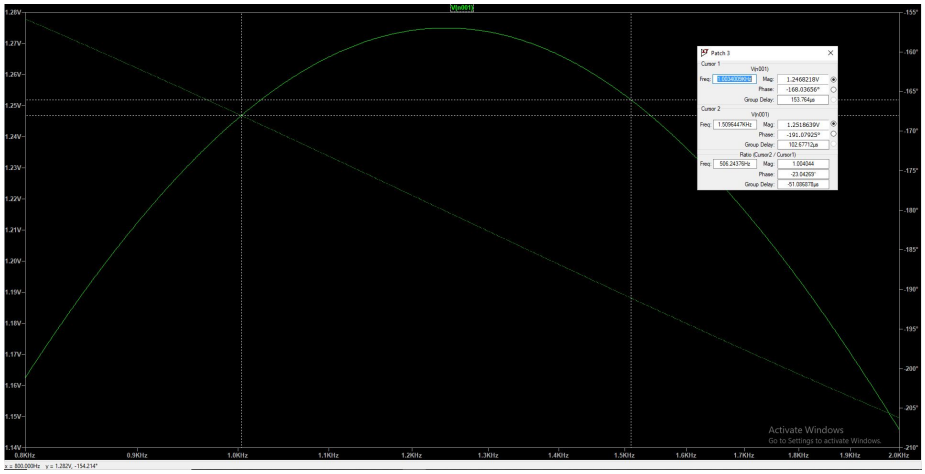


Figure 7: Simulation of Midrange Filter

In Figure 6, we see that cutoff frequency that we want (225 Hz and 450 HZ) is about (350 mV), which is where we would be setting our threshold to detect. There will be a little bit of error as about 500Hz gets included within

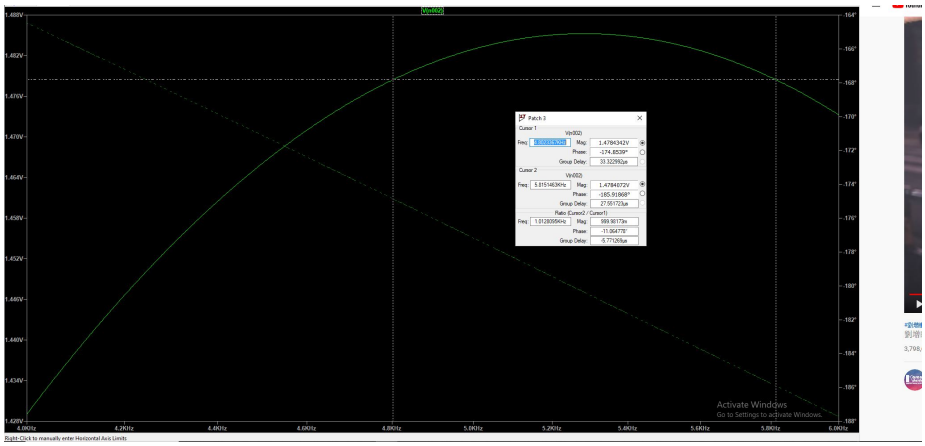


Figure 8: Simulation of Treble Filter

the range, but that should not affect the overall result as it will definitely not include frequency value that are not close to the thresholds.

In Figure 7, we see that the cutoff frequency that we want (1000Hz - 1500Hz) about 1.24-1.25V. Which is the threshold voltage we will be setting for our design. As stated with the Bass filter, there will be a little error around the threshold and probably, about $\pm 5\%$ around the threshold.

In Figure 8, we want the range to be around 4800Hz to 5500Hz. From the simulation, we see that the range is 4800Hz and 5800Hz. It is off due to we want to hold the voltage gain under 3, and thus we do now want it to be super high and make it hard to set threshold for our Arduino program.

7 Constructed Device

The end result is shown in Figure 9. Detail of the breadboard is shown in Figure 10. It is unfortunately messy due to the lack of time needed to refactor and tidy it up. Details are given in the figure captions.

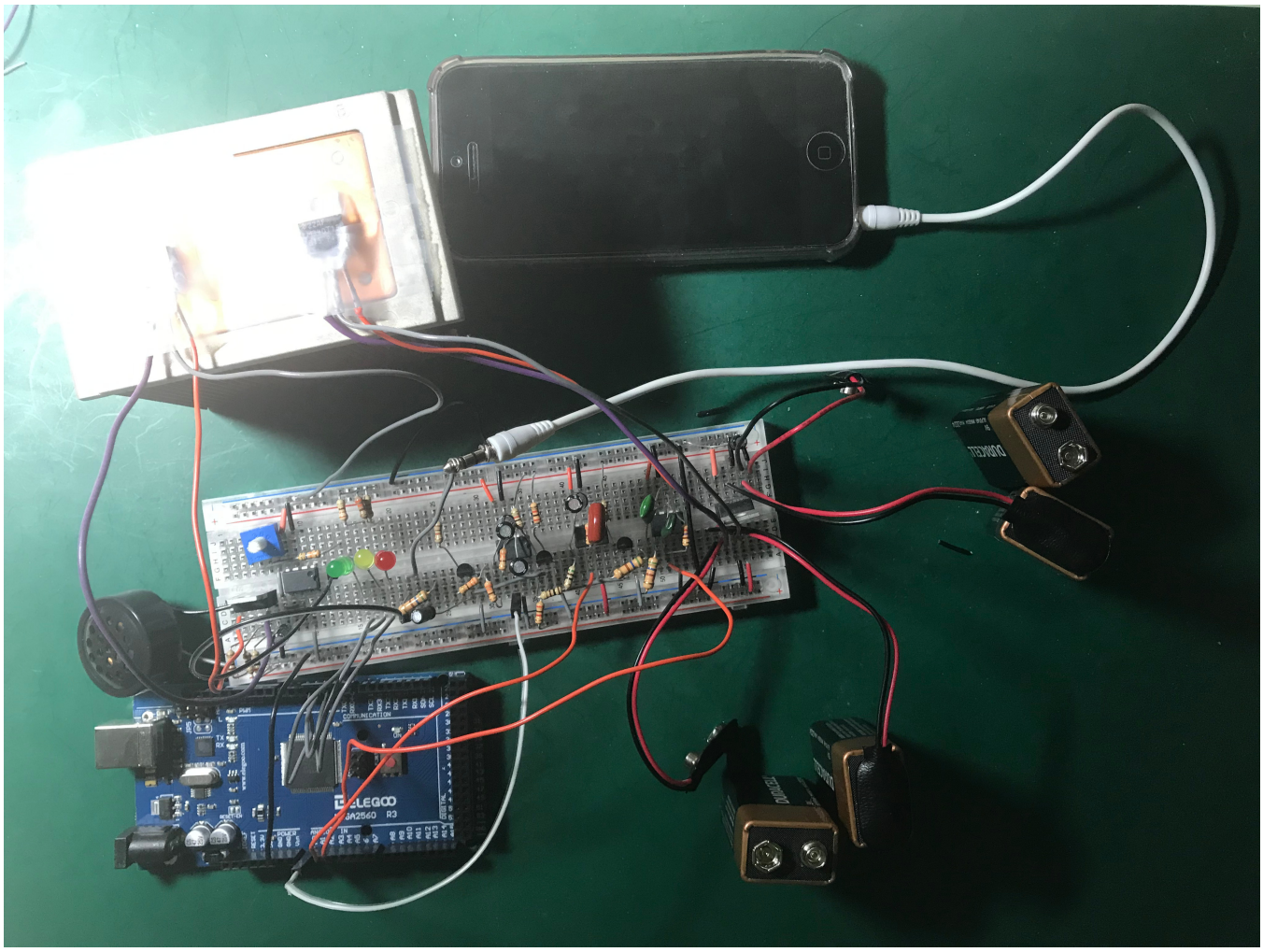


Figure 9: Construction. Notes on the non-breadboard components, clockwise from top left: The TIP42 and LM7815 are (Scotch-taped) attached to a CPU heatsink from an old PC. An iPhone 5 was used as the audio source. Four 9V batteries were used to power the power rails via voltage regulators. The Arduino takes as inputs the (analog) filter outputs and outputs (digital) LED signals. The speaker is an $8\,\Omega$ speaker of unknown wattage.

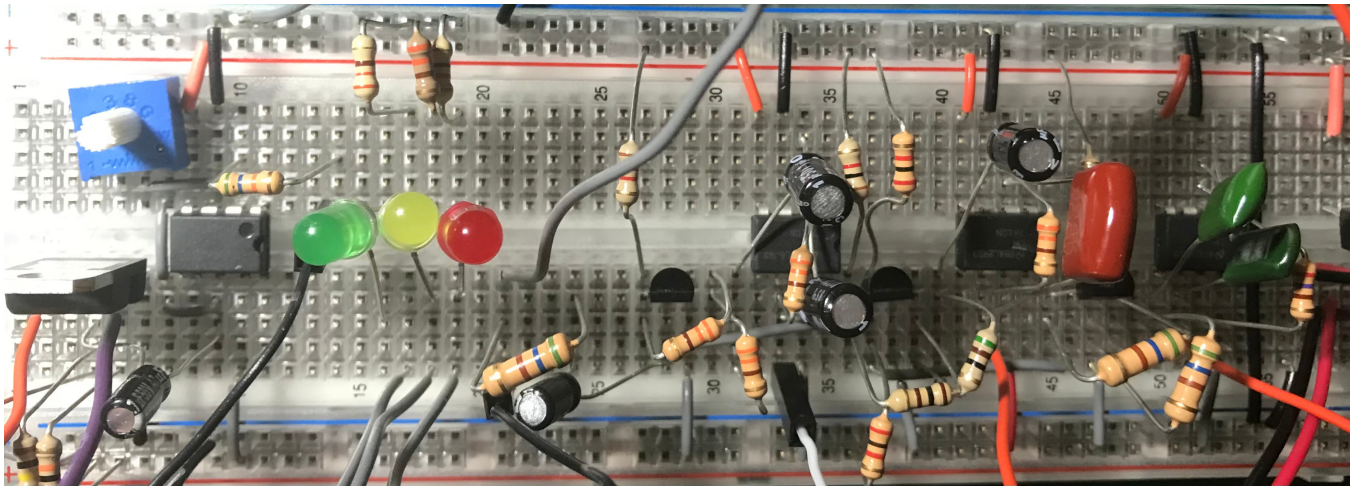


Figure 10: Detail of breadboard in construction. The top rails are GND and VEE; the bottom rails are GND and VCC. On the bottom left, the power amplifier (TIP41 is shown, TIP42 is on the copper heatsink). On the top left (potentiometer and LM741) is the volume control. The input to the volume control (top, row 14) is connected via black jumper cable to the audio input (bottom, row 21), which is also the input to the filters. The three LEDs are used for the color organ output and are triggered by the Arduino via the gray wires on the bottom. The parts to the right of row 55 are the power regulator circuits. The remaining circuits are the three filters: each contains a buffer stage (2N3904 emitter-follower) followed by an active bandpass filter (LM741 op-amp with two capacitors and two resistors – see the schematic diagrams for exact values). There is only one biasing resistor pair on row 36 (two 2 kΩ resistors between VCC and VEE; all three buffers share this same buffer, by connecting their bases together, current-mirror style. The outputs of the filters are jumped to the Arduino analog pins.

8 Evaluation of Constructed Device

The two lower-frequency filters perform decently. This means that, assuming no changes in environmental factors (e.g., temperature) and music amplitude, the filters and the thresholding cause the LEDs to light when the audio is in the correct frequency range. We tested on a YouTube video[5] that performed a frequency sweep at constant amplitude under a short period of time, which fulfills most of these assumptions. This is sufficient for our academic curiosity; however, it would not be very good in a practical setting. A professional color organ should be more robust to noise, changes in music amplitude, changes in environmental factors, and multiple tones.

However, we were unable to get the highest (treble) filter to light the LED at any threshold. We ran low on time and were unable to troubleshoot this. Given that the other two work, we assume this is some fault of our building rather than some fundamental flaw.

There is not much to evaluate about the speaker: the sound is good but it is very power inefficient. It would be a good idea to study the common topologies of the different power amplifier classes before a second attempt at building a power amplifier.

9 Conclusions

The implementation of the project was a lot more difficult than what we expected, as we see that there are problems regarding our third filter on implementation (no detection).

Another key problem about our project is that it should work if the input signal has varying amplitude (mixed volume) as our threshold system using the Arduino is very sensitive to input amplitude. For future work, we would try to make the system robust to any amplitude of audio (perhaps by normalization) and other potential abnormalities in the noise.

Lastly, we did not implement threshold using analog logic, instead doing it with an Arduino. We believe that using a comparator would be better, so that is another thing that we should look into for further implementation.

[5] `https://www.youtube.com/watch?v=qNf9nzvnd1k`

ECE394 – Lab 5-1: Dynamic Curve Tracer

Jonathan Lam

March 2, 2021

The following images follow the results of Lab 5-1 from *The Student Manual for The Art of Electronics*.

1 Setup

We wish to view the I/V characteristics of a device on an oscilloscope. The general idea is to use one channel of the oscilloscope to measure the voltage over the device, and the other channel to measure the voltage over a resistor in series with the device. Since the resistor has a linear I/V characteristic, the second channel is also a measure of current.

We can use the X-Y mode of the oscilloscope to display the channel 2 output (I) as a function of the channel 1 output (V).

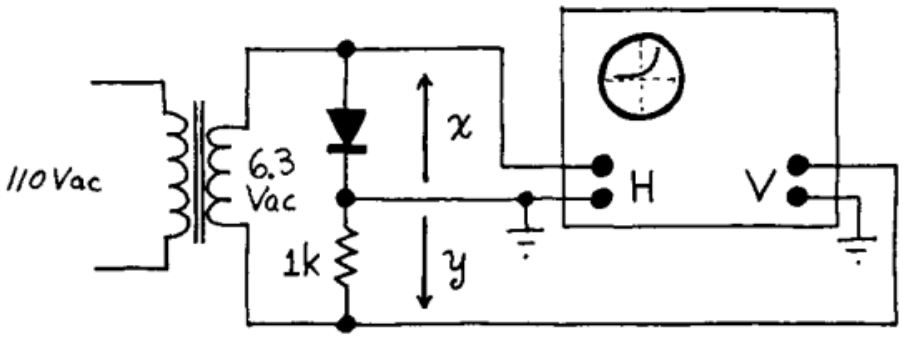


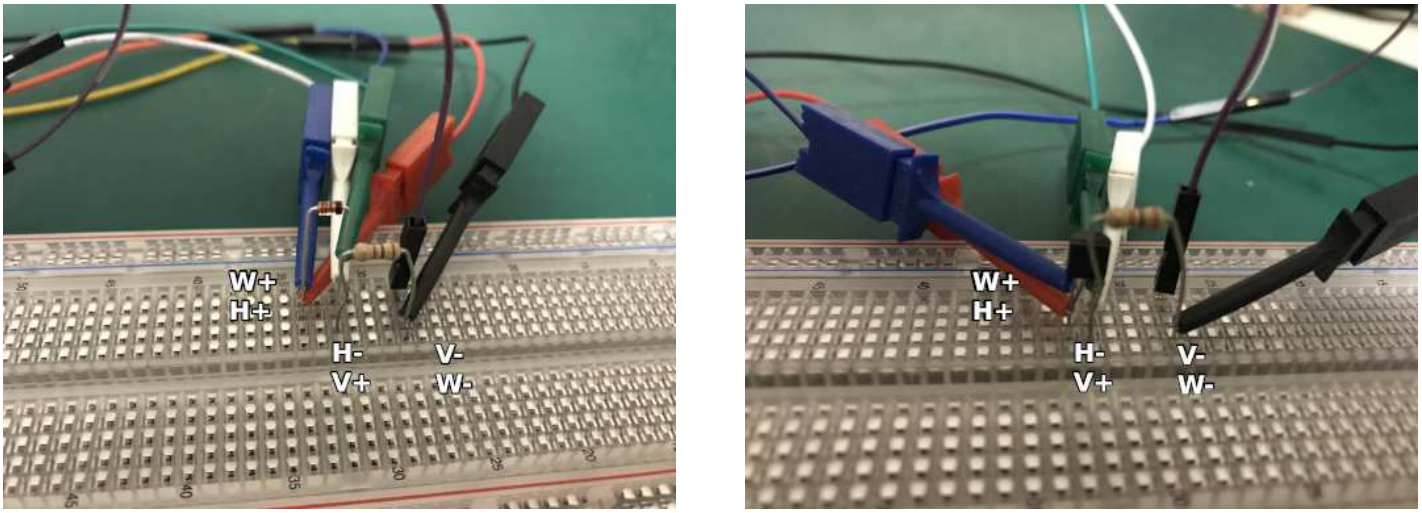
Figure 1: Dynamic Curve Tracer Schematic (source: student manual)

The student manual shows the use of a transformer so that we can set the ground to the central node. However, we can simply use a function generator to power the circuit if we have differential probes. (We simulate the transformer in LTSpice but use a function generator and differential probes in the built circuit.)

As the student manual states, we hope to show the (nonlinear) I/V curve of a diode or transistor. The Ebers-Moll model gives the I/V characteristic:

$$I = I_S \left(\exp \frac{V_{BE}}{V_T} - 1 \right)$$

1.1 Circuit setup



(a) 1N914 Setup

(b) 2N3904 Setup

Figure 2: Built circuit setup

In both setups, the function generator leads W+ and W- are connected on the ends. The first scope channel is connected across the device under test (across the diode, or across the BE-junction). For the transistor, the collector and base are tied together. The second scope channel is connected across the resistor.

Since the resistor is 1kΩ, the voltage value has to be divided by 1000 to get the correct current value (or the value can be taken in units of milliamps).

2 Part A. Diode

2.1 Forward-active diode

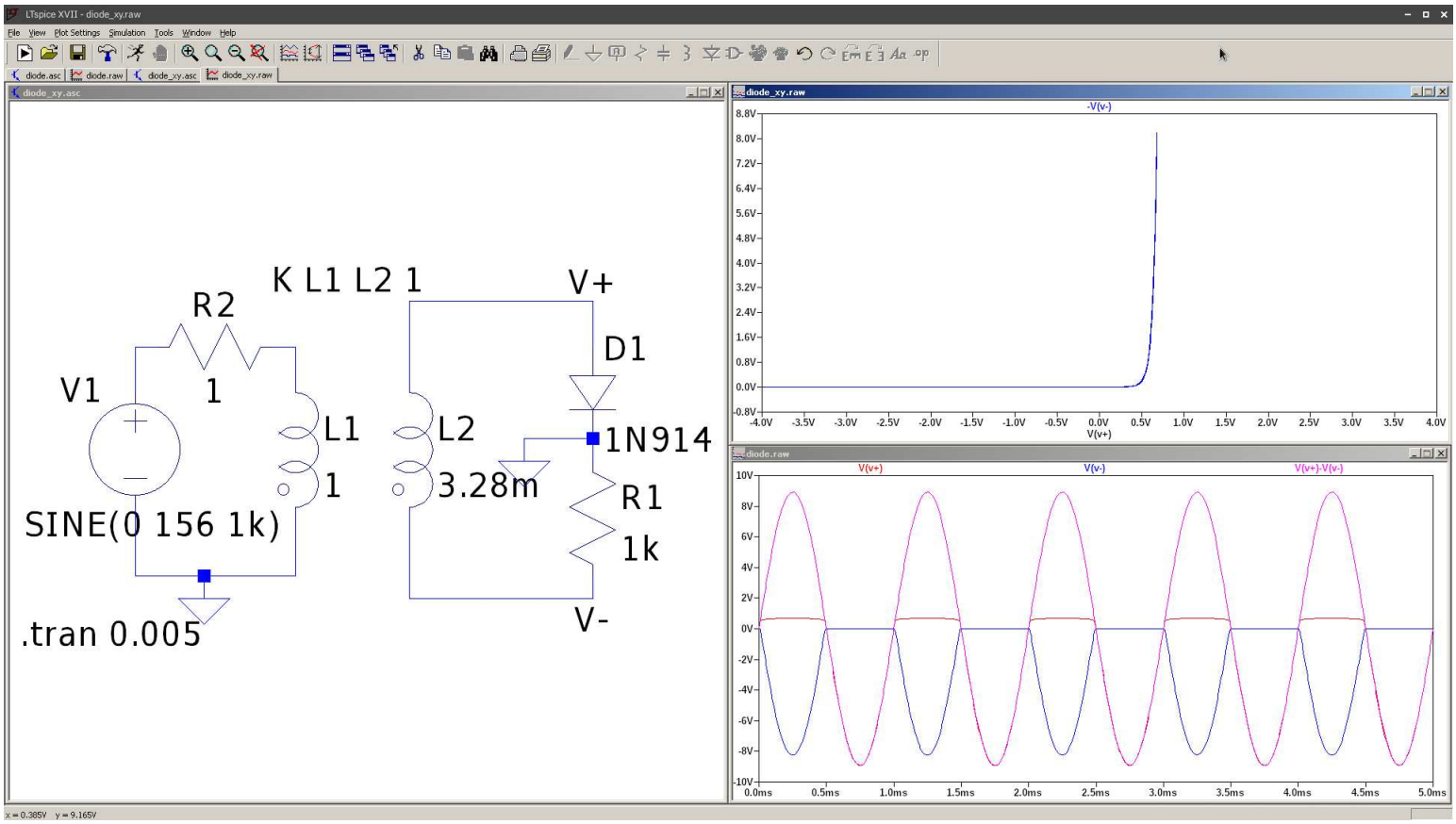


Figure 3: 1N914 Diode LTSpice simulation

The handbook notes that the slope of the V/I curve should be roughly 100mV/decade. This value can be derived from the Shockley diode equation:

$$I_D = I_S \exp \frac{V_D}{nV_T}$$

$$V_D = nV_T \ln \frac{I_D}{I_S} = V_T(\ln I_D - \ln I_S)$$

We want our result w.r.t the I_D on a logarithmic scale (in units of decades):

$$V_{BE} = nV_T \left[\frac{\log_{10} I_D}{\log_{10} e} - \ln I_S \right]$$

$$\frac{\partial V_D}{\partial \log_{10} I_D} = n \frac{V_T}{\log_{10} e}$$

At room temperature, we have $V_T \approx 26\text{mV}$, and the non-ideality factor for the 1N914 (given by LTSpice) is 1.752, so the slope has a value of 105mV/decade.

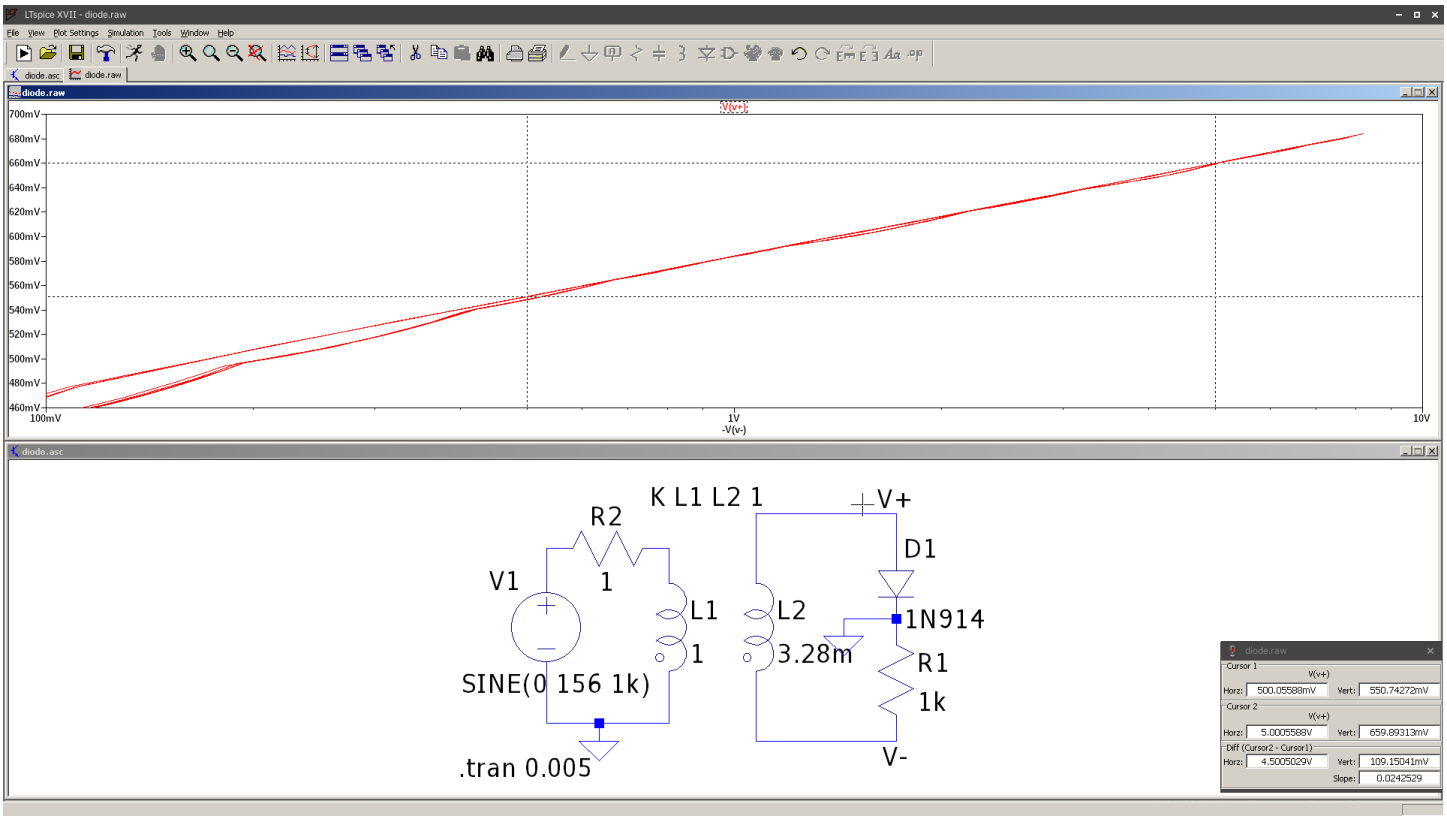


Figure 4: 1N914 Diode LTSpice simulation: showing V as a function of $\log_{10} I$; $\Delta V/\Delta I = 109.15\text{mV/decade} \approx 100\text{mV/decade}$

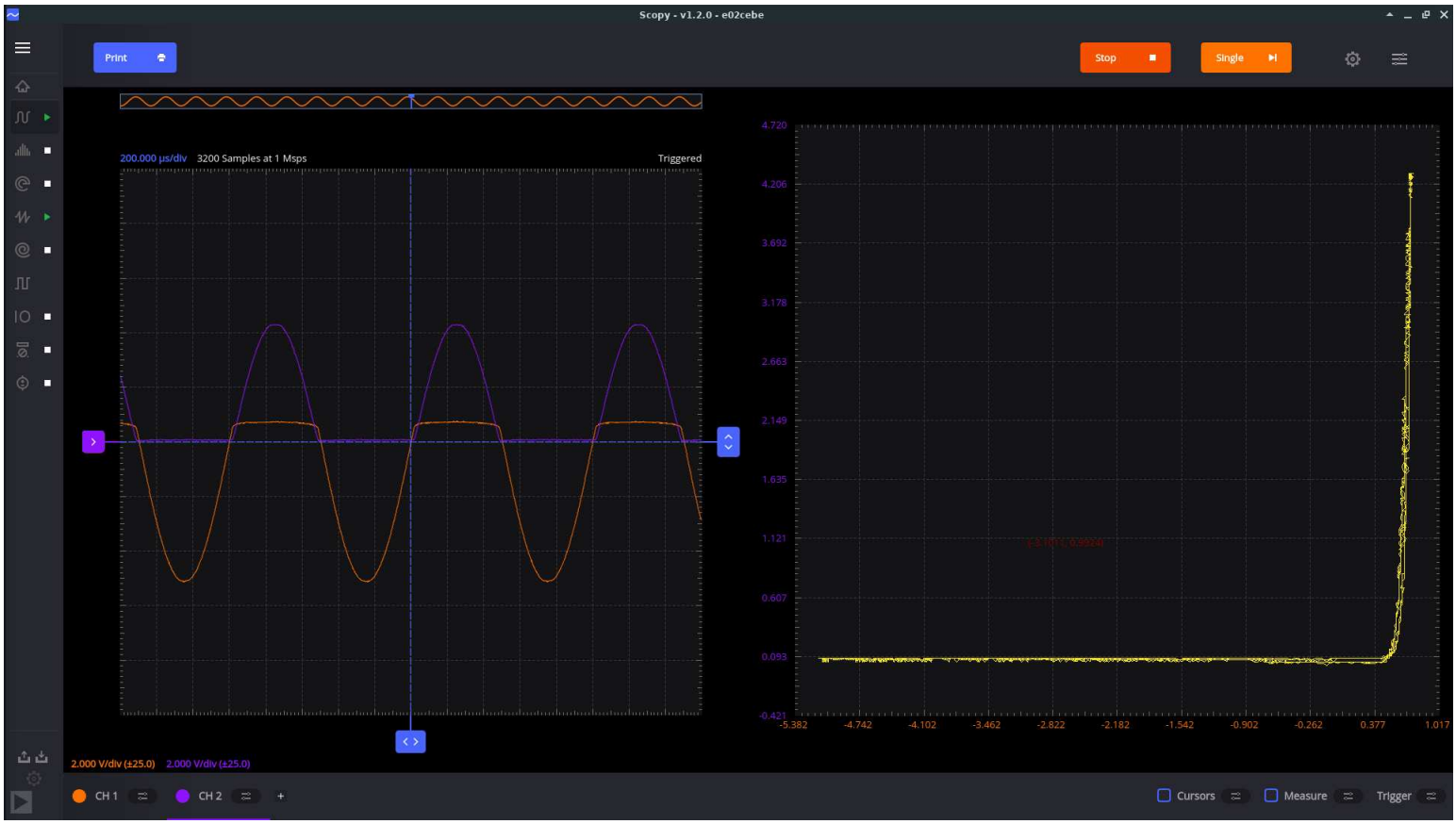


Figure 5: 1N914 Diode I/V characteristic

For this plot, the theoretical results are shown as blue points taken from the X-Y plot on Scopy. The X-axis is voltage (V) and the Y-axis is current (mA). The theoretical equation uses a value for I_S that was found on the Internet for the 1N914. I forgot to add the non-ideality factor, which may account for the slight discrepancy near the origin.

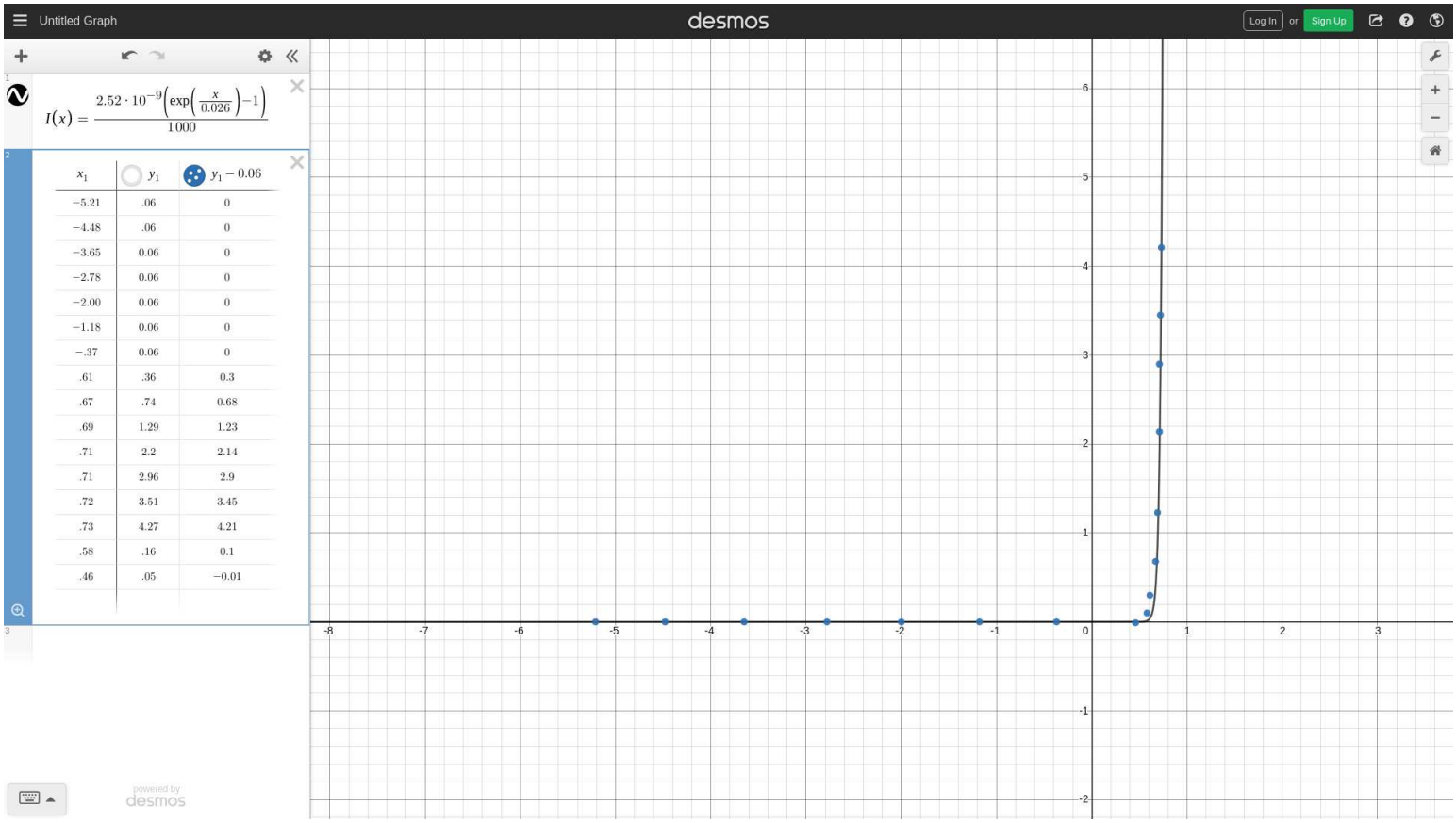


Figure 6: 1N914 Diode I/V Experimental vs. Theoretical

2.2 Reversed diode

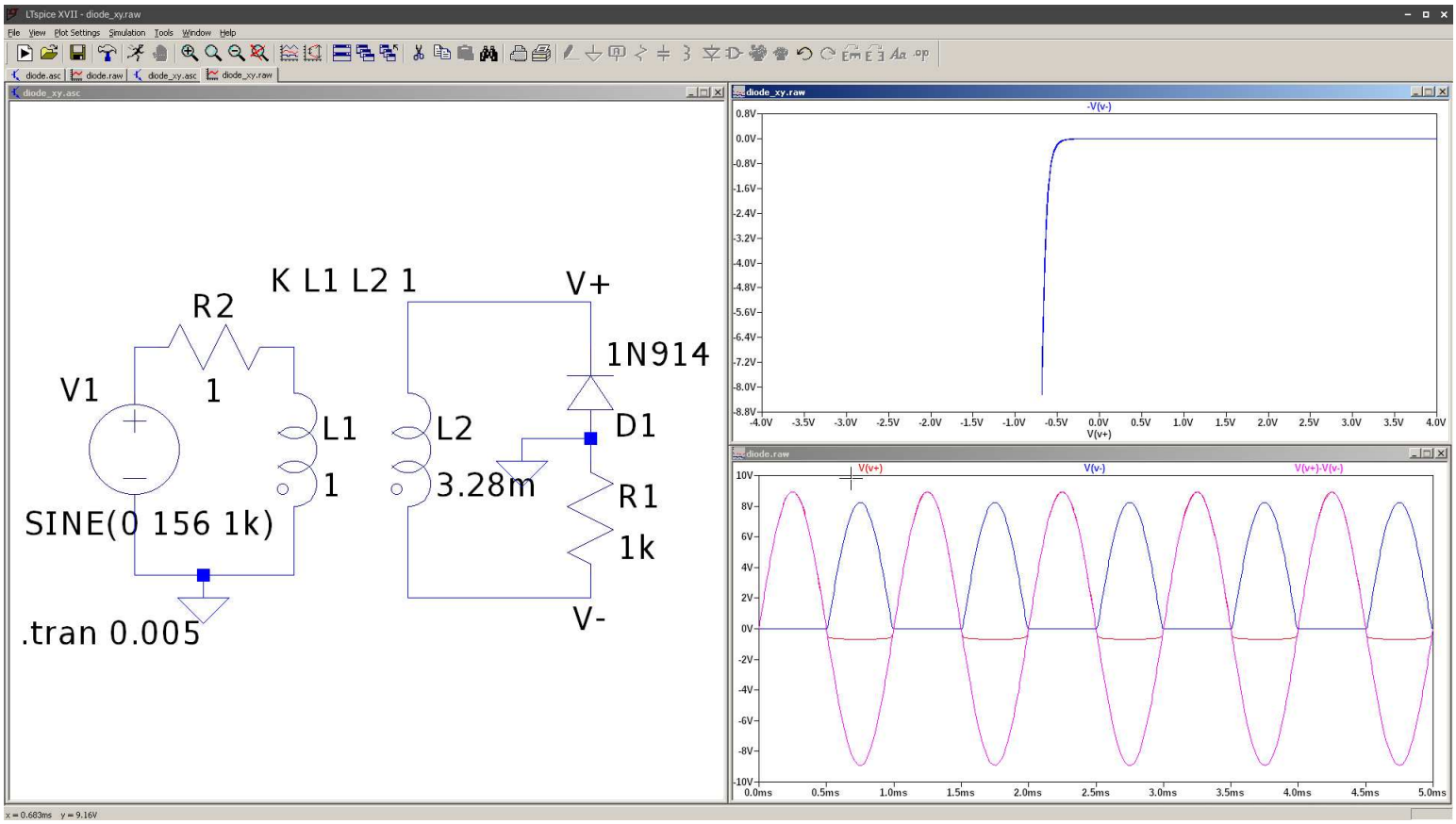


Figure 7: 1N914 Diode Reversed LTSpice simulation

2.3 Zener diode and breakdown

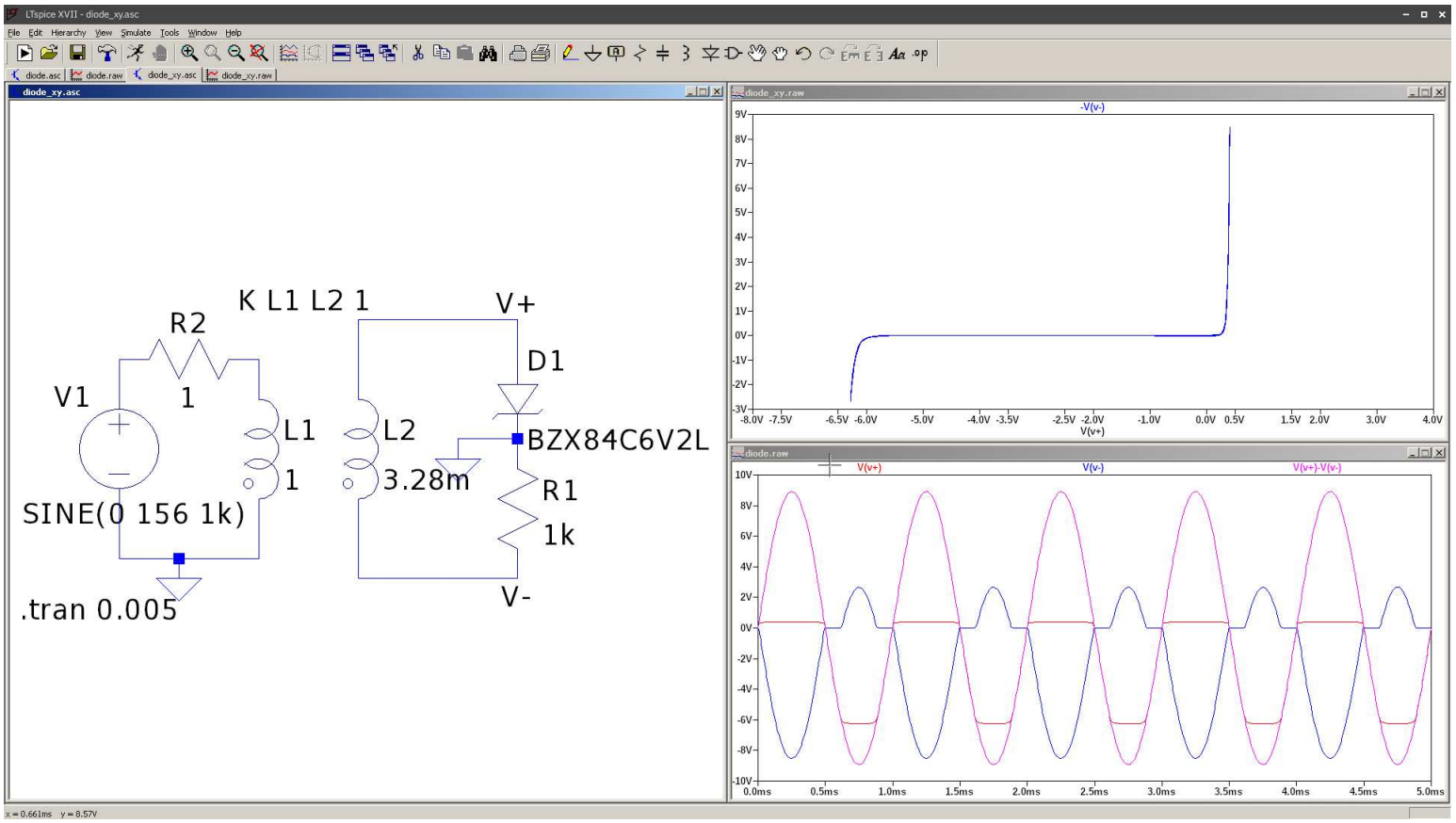


Figure 8: BZX84C6V2L Zener Diode LTSpice simulation ($V_{BD} = 6.2$V)

3 Part B. Transistor

The 2N3904 datasheet states that the breakdown voltage of the 2N3904 is 6V, but LTSpice does not support the breakdown voltage of the BE-junction of a BJT.

3.1 Without protection diode

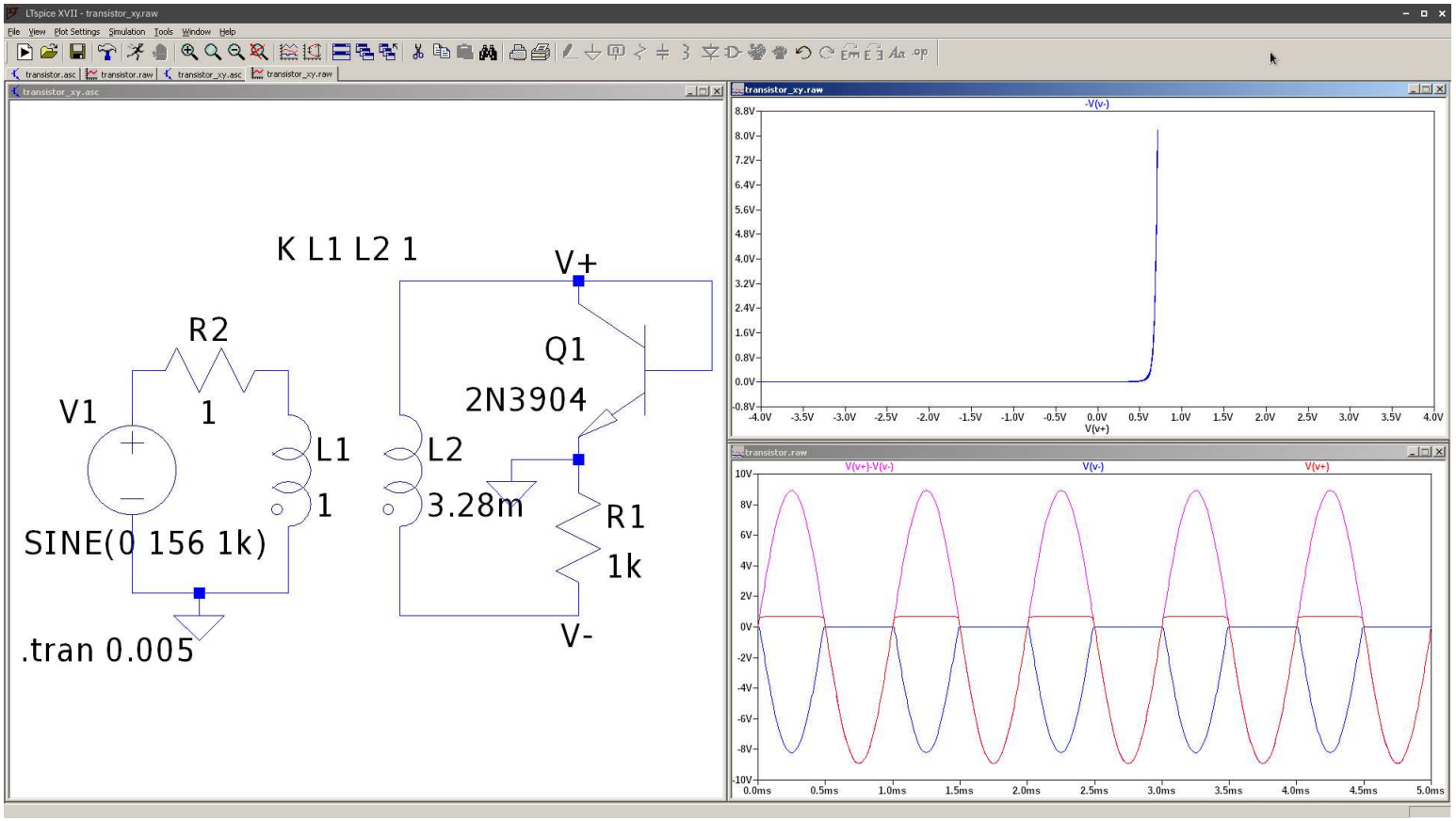


Figure 9: 2N3904 NPN LTSpice simulation

The handbook notes that the slope of the V/I curve should be roughly 60mV/decade. This value can be derived from the Ebers-Moll equation:

$$I_C = I_S \exp \frac{V_{BE}}{V_T}$$

$$V_{BE} = V_T \ln \frac{I_C}{I_S} = V_T(\ln I_C - \ln I_S)$$

We want our result w.r.t the I_D on a logarithmic scale (in units of decades):

$$V_{BE} = V_T \left[\frac{\log_{10} I_C}{\log_{10} e} - \ln I_S \right]$$

$$\frac{\partial V_D}{\partial \log_{10} I_C} = \frac{V_T}{\log_{10} e}$$

At room temperature, we have $V_T \approx 26\text{mV}$, so the slope has a value of 59.9mV/decade.

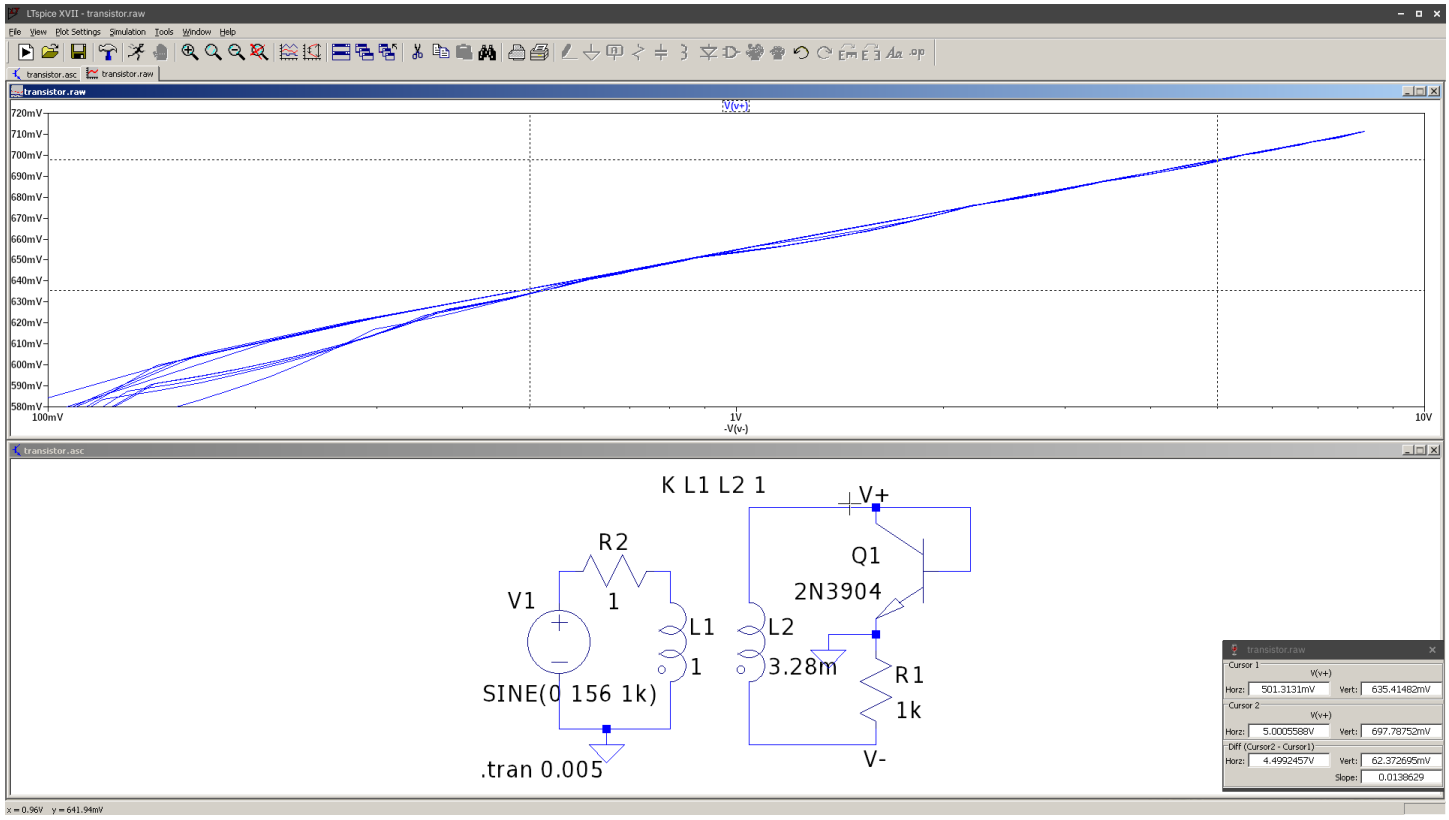


Figure 10: 2N3904 Diode LTSpice simulation: showing V as a function of $\log I$; $\Delta V/\Delta I = 62.37\text{mV/decade} \approx 60\text{mV/decade}$

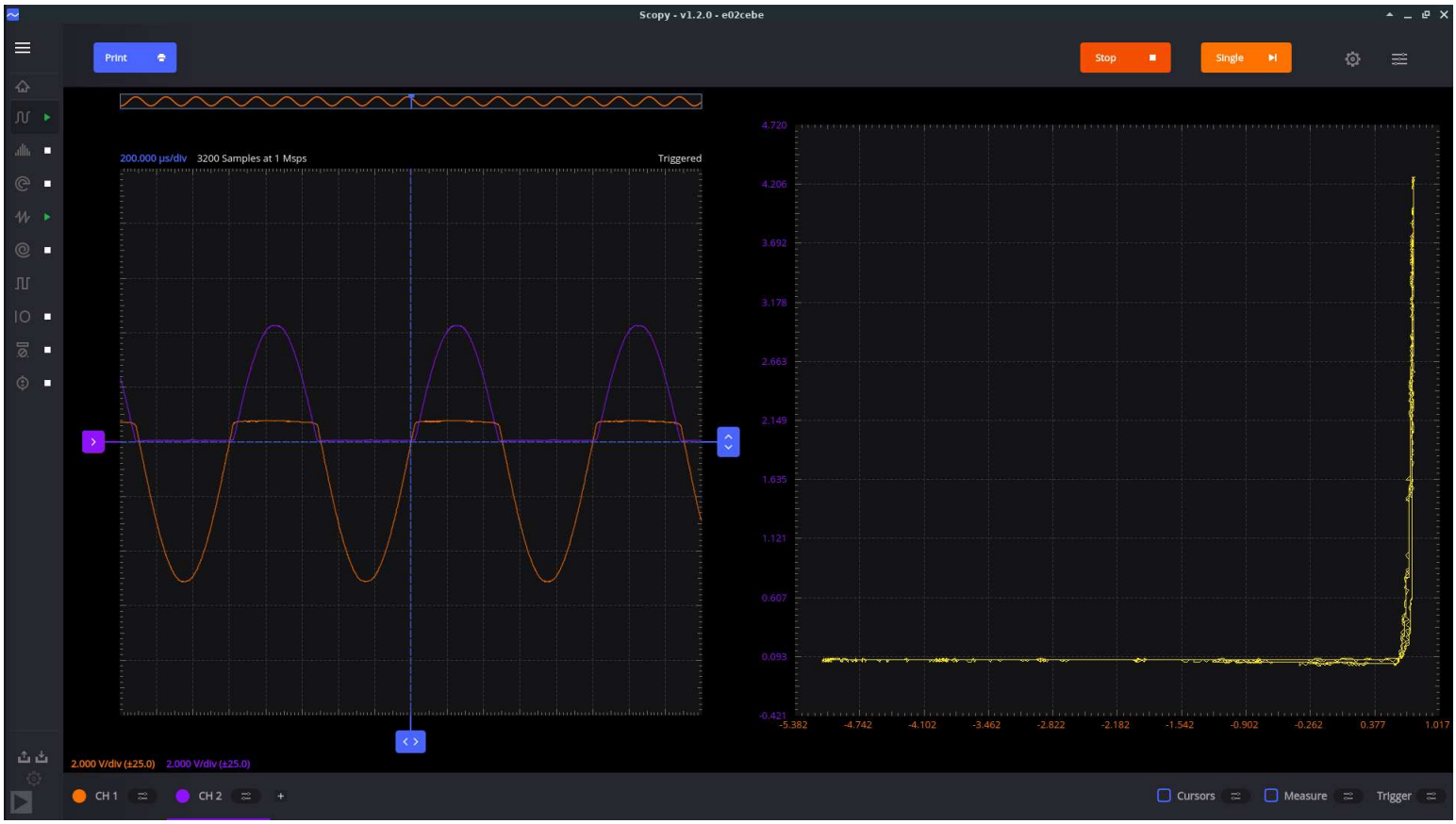


Figure 11: 2N3904 NPN I/V characteristic

The X-axis is voltage (V) and the Y-axis is current (A). This was done a little differently than for the diode – I calculated a value for I_S using a sample quiescent point, rather than using one off of the Internet.

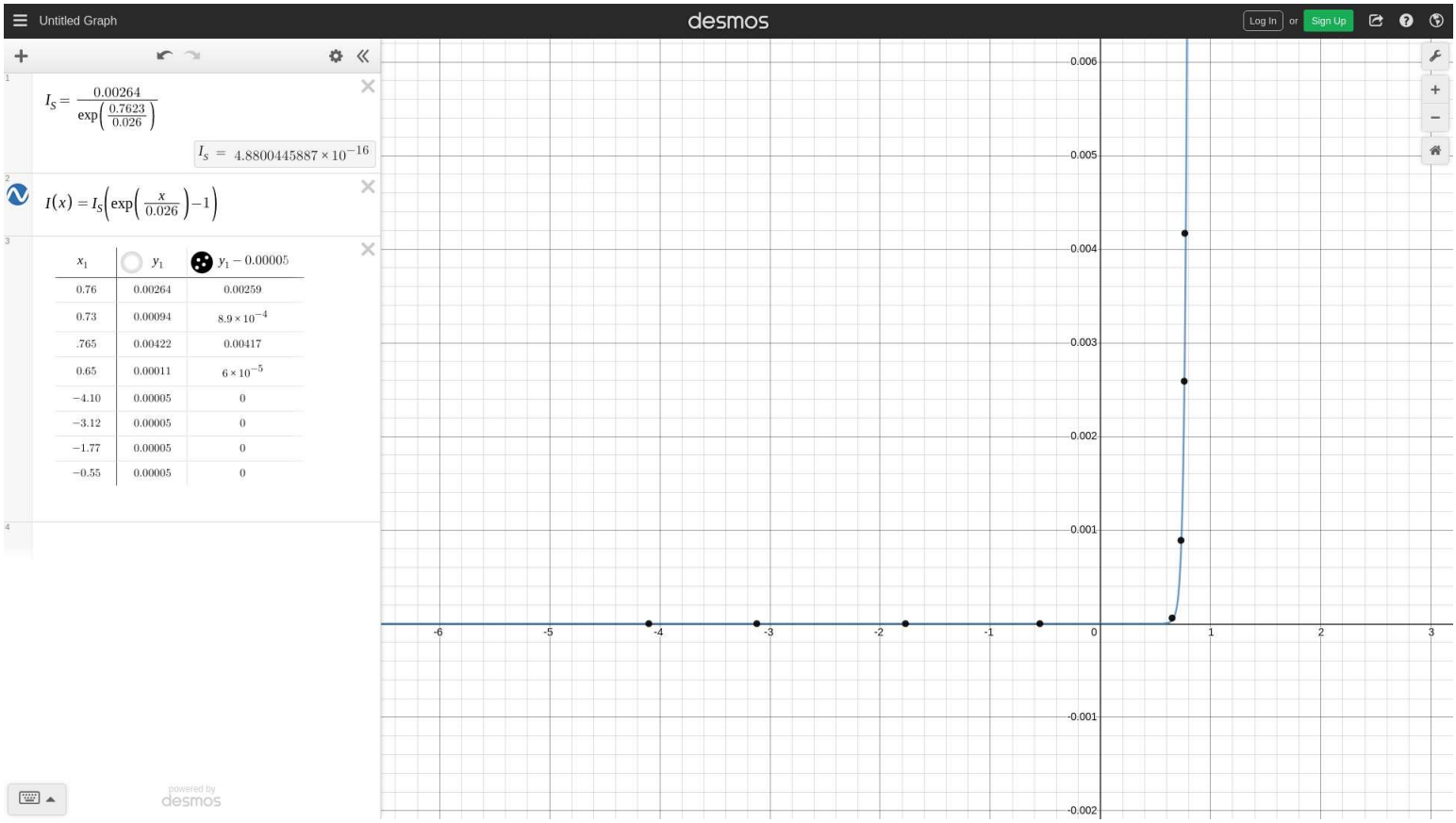


Figure 12: 2N3904 NPN I/V Experimental vs. Theoretical

3.2 With protection diode

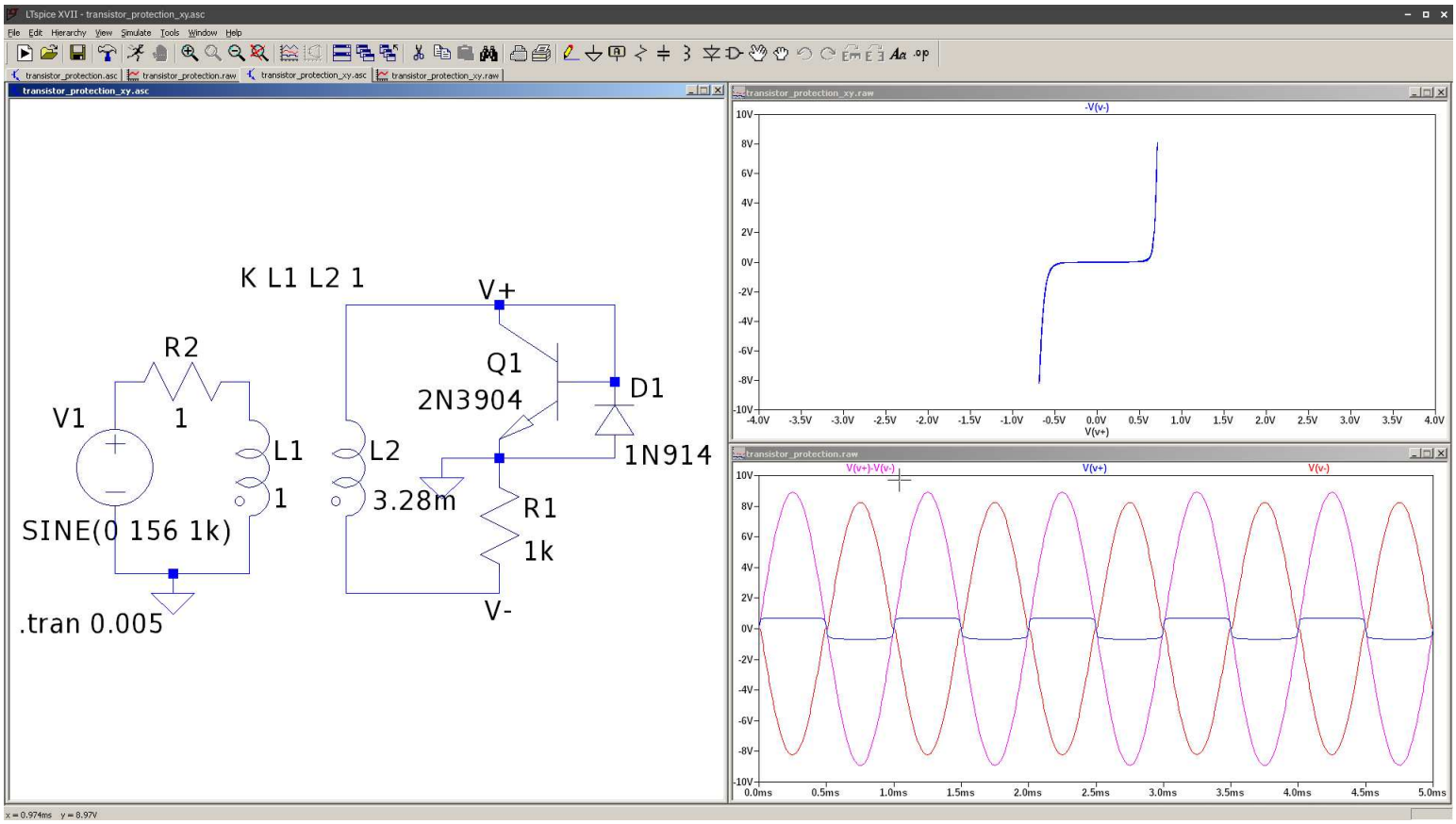


Figure 13: 2N3904 NPN LTSpice simulation with protection diode

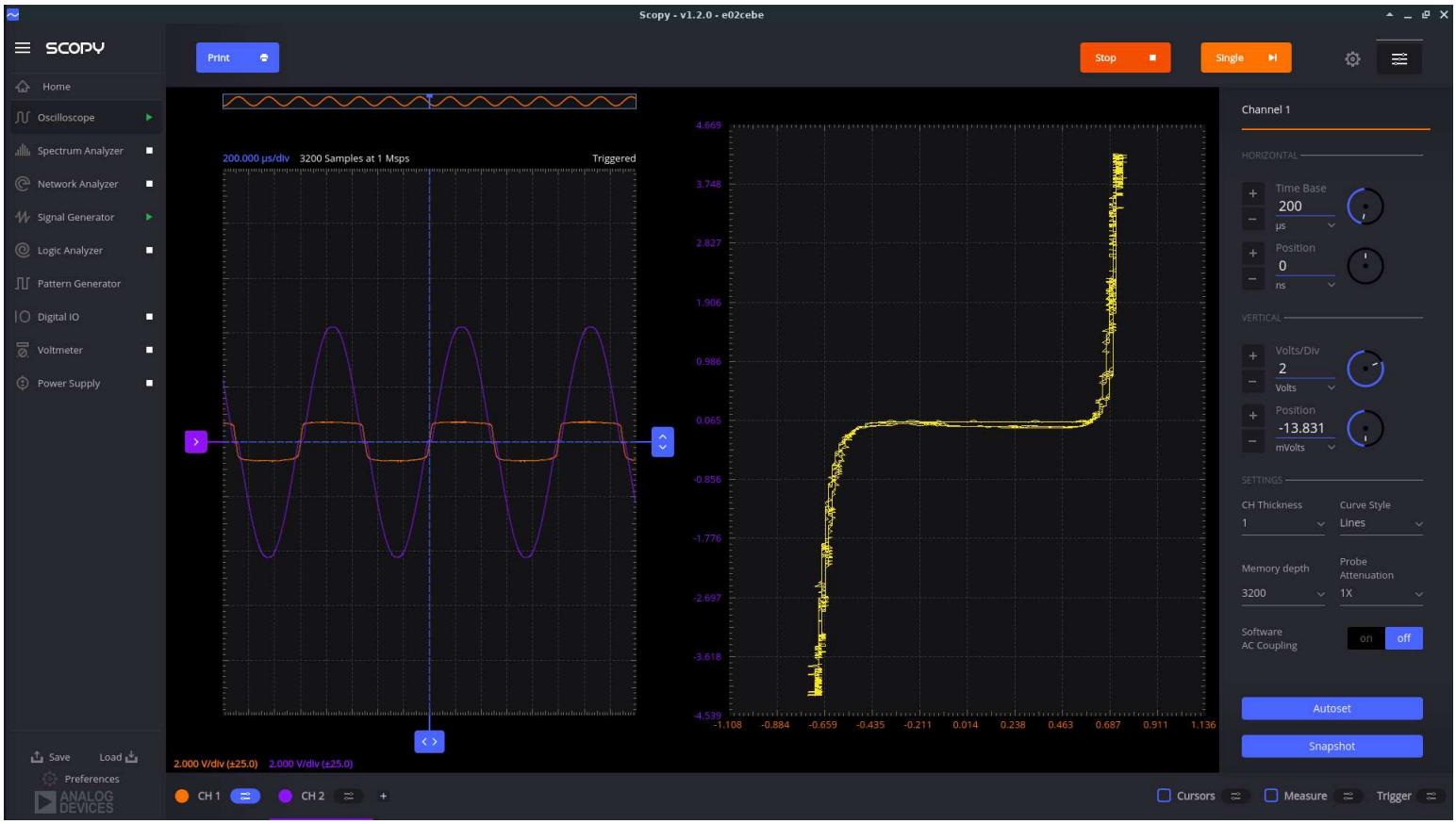


Figure 14: 2N3904 NPN with Protection Diode I/V Characteristic

ECE394 – Lab 8-7

Jonathan Lam

March 30, 2021

1 Introduction

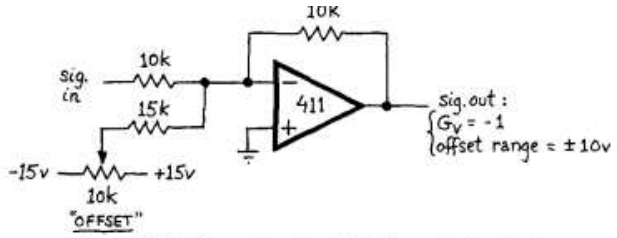


Figure L8.14: Summing circuit: DC offset added to signal

Figure 1: Textbook schematic of adder circuit. "The circuit in the figure sums a DC level with the input signal. Thus it lets you add a DC offset to a signal. (Could you devise other op amp circuits to do the same task?)"

2 Theoretical analysis

To simplify the analysis, we treat the offset voltage simply as a (variable) DC voltage source, and denote it V_{off}. This disregards the resistance of the potentiometer, which is small enough that we approximate it away.

Then we have an input circuit to an inverting op-amp circuit, which can further be simplified by viewing its Thevenin equivalent. This is:

$$V_{\text{th}} = V_{\text{in}} - (V_{\text{in}} - V_{\text{off}})\frac{R_1}{R_1 + R_2}$$
$$R_{\text{th}} = R_1 || R_2$$

What is important is that V_{th} is related linearly to V_{off}. This then gets passed through a inverting or non-inverting amplifier, which have gains of

$$A_{V_{\text{invert}}} = -\frac{R_F}{R_{\text{th}}}$$
$$A_{V_{\text{noninvert}}} = \frac{R_2 + R_F}{R_2}$$

See Figure 2 for a sketch of this analysis.

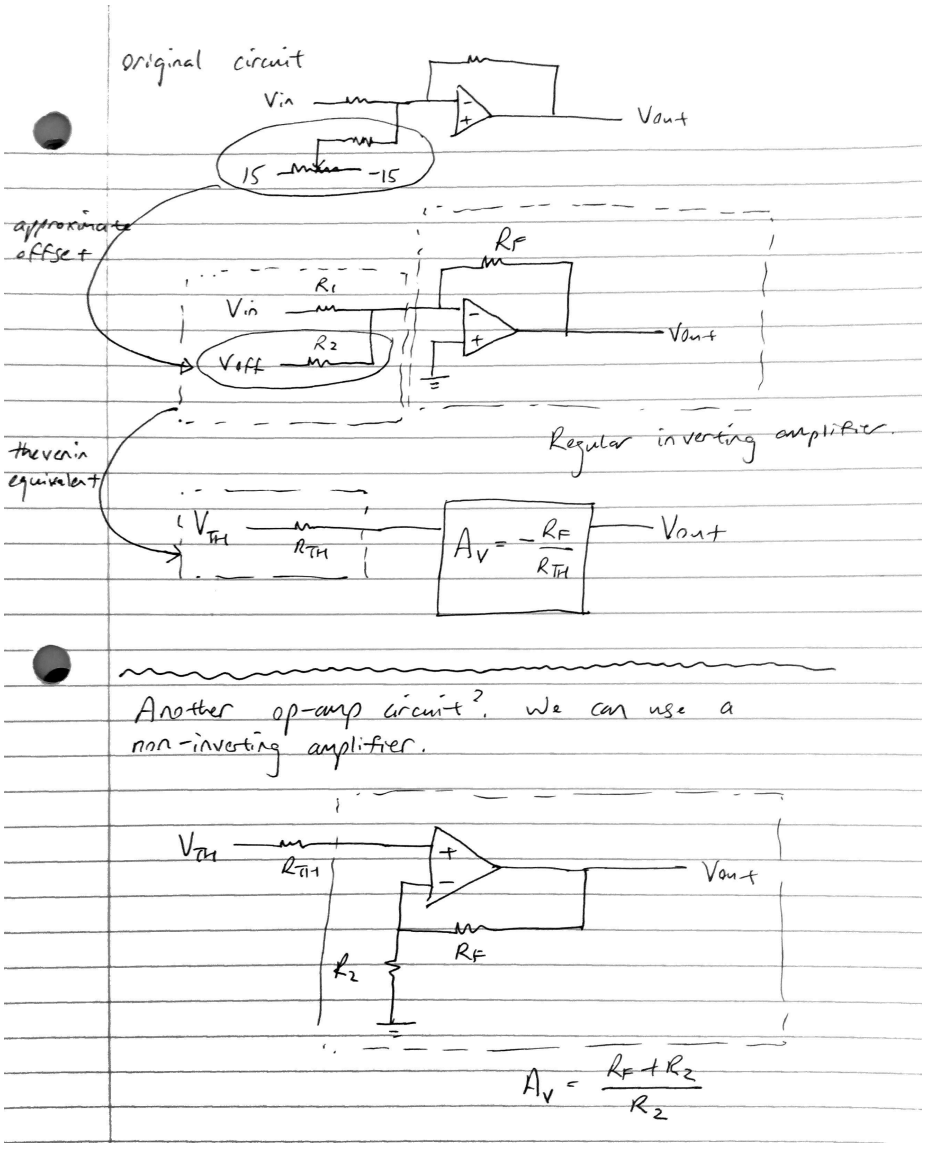


Figure 2: Theoretical analysis and simplifications made.

3 Circuit implementation

A LM741 was used in place of a LM411 because the LM411 was not available.

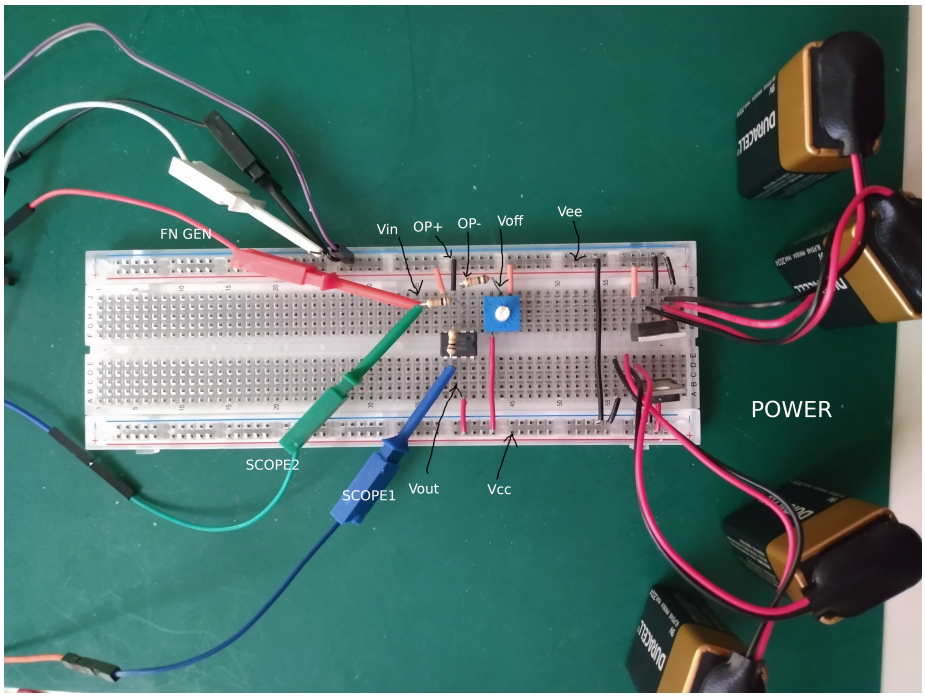


Figure 3: Implementation of circuit on the breadboard.

4 Results

See this video for a demonstration of the DC offset.

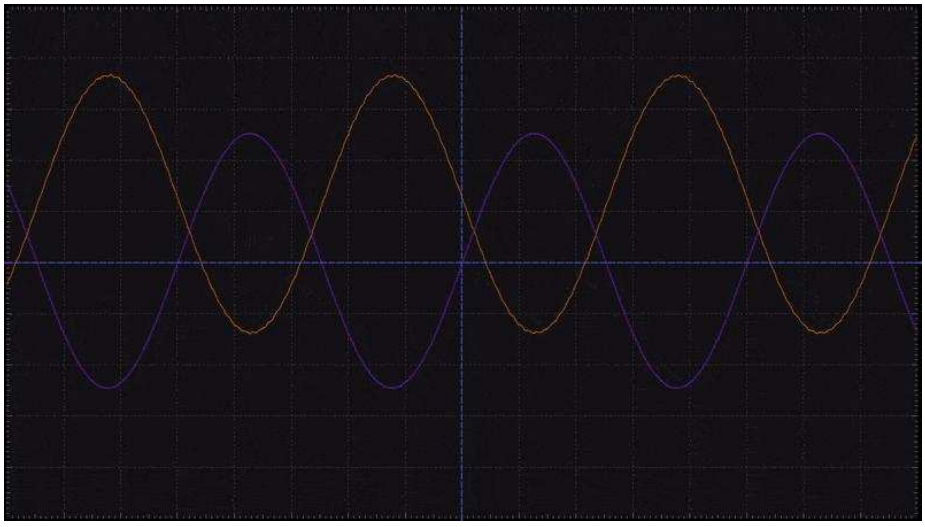


Figure 4: Screenshot of Scopy showing input signal and DC-offset output signal

5 LTSpice implementation

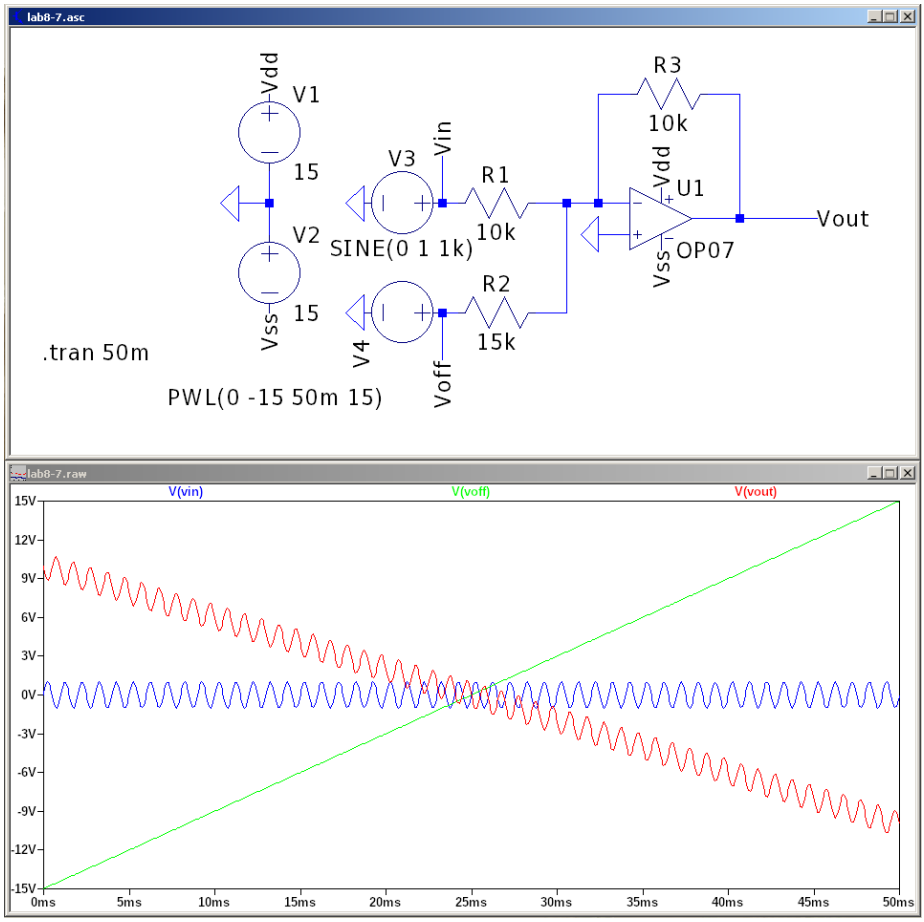


Figure 5: LTSpice inverting summing amplifier

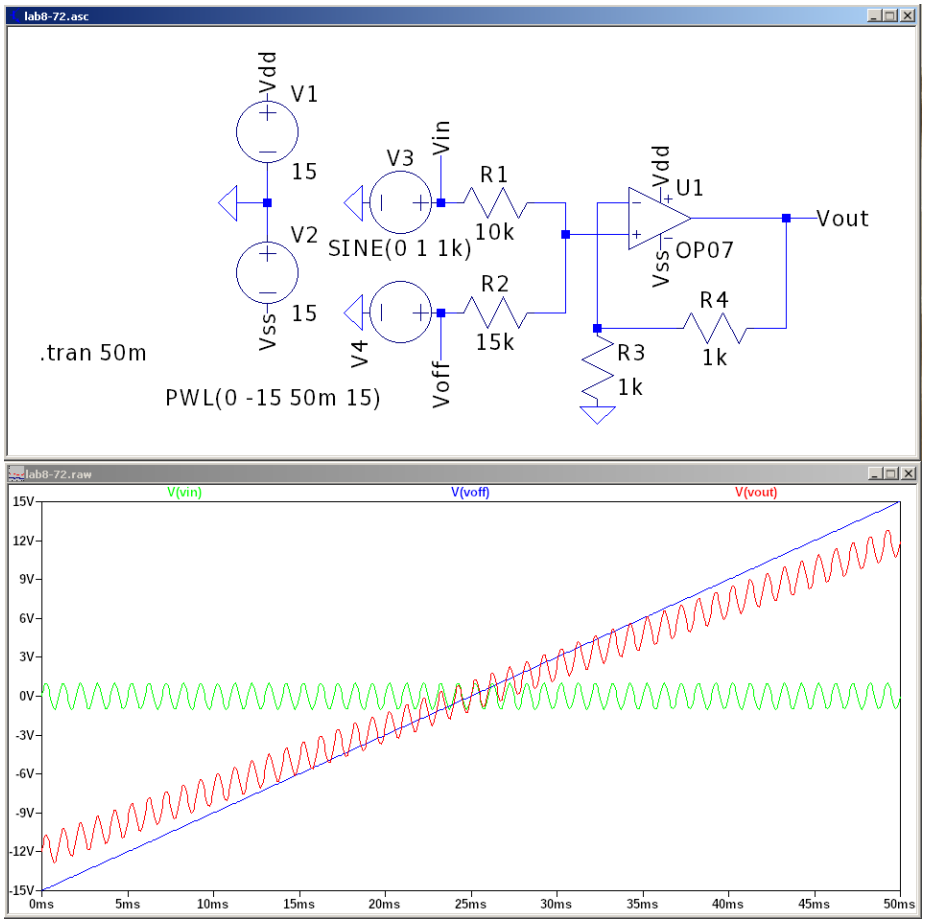


Figure 6: LTSpice non-inverting summing amplifier

Edge Detection - Canny Filter Implementation

Jonathan Lam

May 10, 2021

1 Abstract

Edge detection is a common and important task in image processing. One of the standard edge detection algorithms is the Canny edge detector, a six-stage algorithm. We implement the the filter in CUDA C++ based on Luo and Duraiswami [1] and a baseline CPU version. This paper describes the theory behind the Canny filter, a naive approach covering all six stages, and several optimizations based on Luo and Duraiswami's work, along with relevant code samples. We evaluate the performance of the different implementations compared to one another on several test images.

2 Introduction

According to the MATLAB documentation[1], "Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision. Common edge detection algorithms include Sobel, Canny, Prewitt, Roberts, and fuzzy logic methods."

We had originally decided to implement the Sobel filter, which comprises of two (a horizontal and vertical) 3×3 differential filters. However, this proved to be too simple for the purposes of this project, so we decided to attempt an implementation of the Canny algorithm, which includes the Sobel filter as the second step.

The Canny filter was developed by John Canny in 1986 [2]. It comprises six steps:

1. turn the image into a single channel (grayscale)
2. applying a smoothing (denoising) filter
3. finding the intensity gradients and directions
4. non-maximum edge suppression (i.e., edge thinning)

[1] `https://www.mathworks.com/discovery/edge-detection.html`

5. double-thresholding the remaining edges

6. edge tracking via hysteresis

The following sections will review the intuition and general method for each step. A visual summary of the steps can be found in Figure 1; this is our reproduction of the example provided on the Wikipedia page for the Canny filter[2].

2.1 Grayscale

Edge detection depends only on the brightness gradients of the image, and not on the color. If we begin with a color image, we must begin by converting it to some grayscale variant. We use the luminance measure.

2.2 Smoothing filter

This is used as a method of edge control. We want to avoid spurious high-frequency components from appearing as noise, while mostly preserving larger (true) edge features. A common choice is a (2D circular) Gaussian blur with a small blur radius. We use a Gaussian blur with standard deviation 2 for most of our experiments, although this depends on the density of the details in the image.

2.3 Finding the intensity gradients and directions

This is where the Sobel-Feldman (Sobel) filter comes in. The Sobel filters are small, separable 3×3 filters that provide us with integral approximations of the horizontal and vertical gradients of the intensity of the image.

$$T_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad T_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

After finding the horizontal and vertical gradients, we find the (2D) image intensity gradient via the following formula:

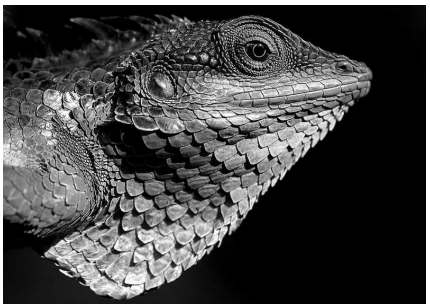
$$|G| = \sqrt{G_x^2 + G_y^2} \tag{1}$$

and we can find the direction of the gradient with:

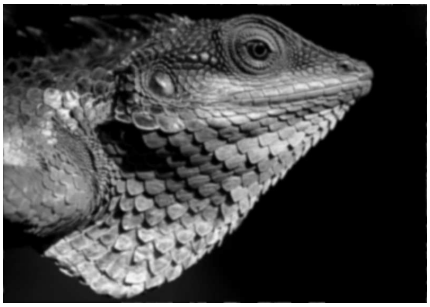
$$\angle G = \arctan \frac{G_y}{G_x} \tag{2}$$

[2] `https://en.wikipedia.org/wiki/Canny_edge_detector`

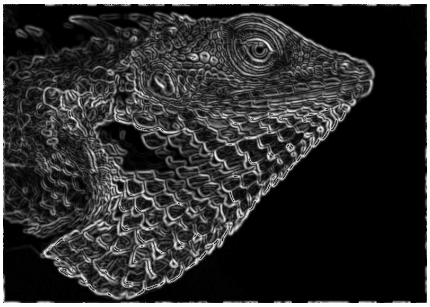
(a) Original image



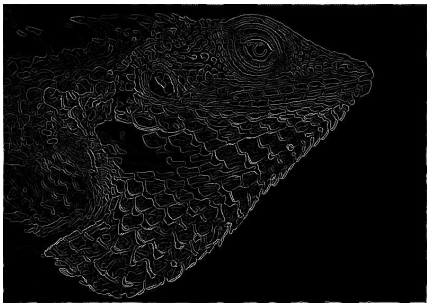
(b) Luminance operator



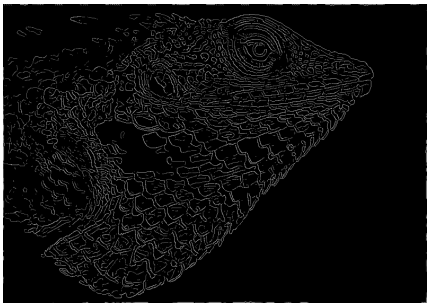
(c) Gaussian blur



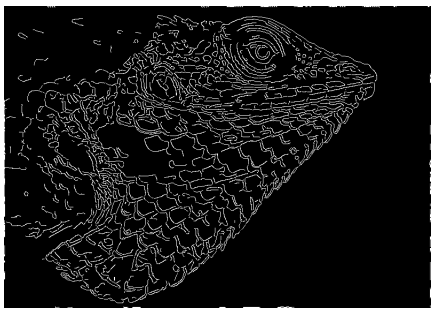
(d) Sobel filter



(e) Edge thinning

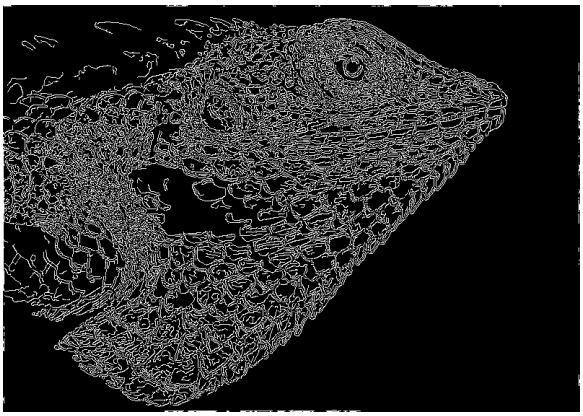


(f) Double thresholding

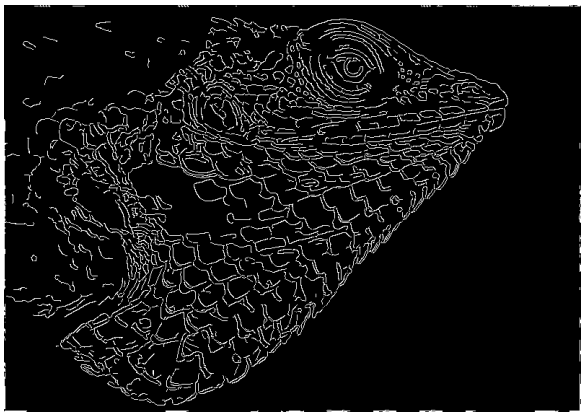


(g) Finished canny

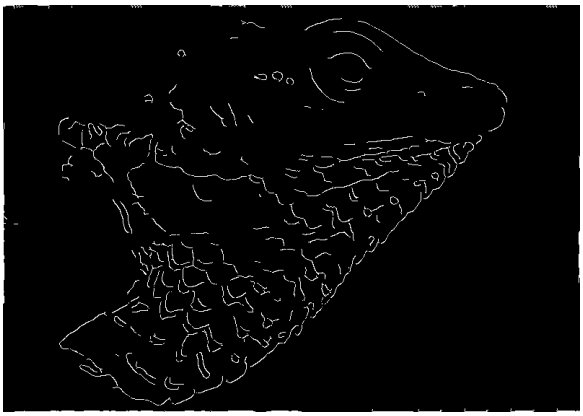
Figure 1: Canny edge detection stages. Zoom in for detail. Babujayan, CC BY 3.0, via Wikimedia Commons.



(a) $\sigma = 1$



(b) $\sigma = 2$



(c) $\sigma = 3$

(d) $\sigma = 4$

Figure 2: Effect of changing blur standard deviation on the same lizard image. The blur size is dependent on the detail density (resolution) of the image. Increasing the blur size will decrease noise as well as the number of detected edges. It may also be necessary to change the thresholds based on the blur size. $\sigma = 2$ was chosen as the default for our tests as that seemed to work fairly well for reasonable resolutions.

2.4 Non-maximum edge suppression

The Sobel filter provides a fairly recognizable magnitude representation, but Canny decided that a more accurate representation of "edges" are represented by the zero-crossings of the second derivatives of the intensity gradients. This step uses the angle calculated in the previous step and determines if a pixel is a local maximum in of the directional intensity gradient (which corresponds to a zero-crossing in the second derivative). Usually, to simplify this calculation, the angle is quantized rounded into one of four angles $\{0, \pi/4, \pi/2, 3\pi/4\}$. (Note that opposite angles are not distinguishable, i.e., 0 and π are considered the same angle.) Values that are not local maxima are zeroed. The visual effect of this is that the edges become thinner.

2.5 Double-thresholding

This step and the next are further used to suppress noisy pixels. We mark remaining pixels above a high threshold as having a strong gradient, and mark pixels between that threshold and a lower threshold as having a weak gradient. Pixels with an intensity gradient lower than the lower threshold are zeroed.

2.6 Edge tracking via hysteresis

All weak pixels that are connected to a strong pixels are labeled strong. This process iteratively until there are no remaining weak pixels connected to a strong pixel. The end result of the Canny filter are the strong pixels, and the remaining weak pixels are zeroed. The intuition behind this step is that the strong pixels are already determined to be part of edges; the weak pixels, however, are more likely to come from noise, and are more likely to come from an edge if they are near other known edge pixels. This should be implemented via some sort of BFS algorithm.

3 Implementation and Optimizations

Each of the above steps roughly corresponds to one CUDA kernel. We will describe the naïve implementation and any optimizations performed for each step.

The CUDA program was implemented and tested using CUDA 9.2 on a GT 740, which has Compute Capability (CC) 3.0, on a Debian 10 computer with an i7-2600 CPU. The limiting factor of the GT 740 is that it has 1GB of VRAM, which caused out-of-memory (OOM) errors for images larger than 16k (15360×8640, or 132MP) – thus, our largest test cases are (roughly) 16k images.

3.1 Image preprocessing

Images were read and written using libpng. The code to process (read, decode, encode, write) the PNG files is courtesy of Guillaume Cottencau[3].

Libpng is (probably) not very efficient and the code is not very flexible, but it was simple enough for our needs.

3.2 Timing

A simple custom timer was written using the `clock()` function from the `<clock.h>` STL header. To prevent asynchronous kernel invocations, `cudaDeviceSynchronize()` was used when benchmarking the kernel elapsed times.

3.3 Grayscale

The grayscale operator is found in Figure 3. This uses the RGB-to-luminance formula from ITU BT.601[4].

$$I = 0.2989R + 0.5870G + 0.1140B \tag{3}$$

The first implementation used floating-point calculations, using this formula literally. Simply eliminating floating-point calculations as in Figure 3 reduced runtime by about 30%. To reduce time even further, we can use even lower precision, as color accuracy is not of utmost importance. Some other integral approximations can be found on the Internet[5].

3.4 Gaussian blurring

The convolution with the Gaussian blur is the slowest kernel. The original naive implementation involves a 2D kernel loaded into global memory and convolved at every pixel. The dimensions of the filter are determined by the blur standard deviation, and is calculated as so:

$$H = \lceil 6\sigma \rceil + 1$$

The (somewhat arbitrary) coefficient of 6 is to ensure that we capture enough of the Gaussian's filter; the +1 is used to make the filter an odd size to simplify calculations. Note that with a (somewhat reasonable) blur size $\sigma = 5$, this means that the filter is $31 \times 31 = 961$ pixels – this means 961 multiplications (and 1922 global memory accesses in the naive version with the filter in global memory) per thread.

The convolution then simply multiples pixels pointwise with the filter. There is a translation happening that causes the convolution to be centered around the image and not the filter (i.e., a "same" padding convolution). By moving

[3] `http://zarb.org/~gc/html/libpng.html`
[4] `https://stackoverflow.com/a/596241`
[5] `https://www.programmersought.com/article/20584662327/`

```
__global__ void toGrayScale(byte *dImg, byte *dImgMono, int h, int w, int ch)
{
        int ind, y, x;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        if (y >= h || x >= w) {
                return;
        }

        ind = y*w*ch + x*ch;

        dImgMono[y*w + x] = (2989*dImg[ind] + 5870*dImg[ind+1]
                + 1140*dImg[ind+2])/10000;
}

// convert back from single channel to multi-channel
__global__ void fromGrayScale(byte *dImgMono, byte *dImg, int h, int w, int ch)
{
        int ind, y, x;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        if (y >= h || x >= w) {
                return;
        }

        ind = y*w*ch + x*ch;
        dImg[ind] = dImg[ind+1] = dImg[ind+2] = dImgMono[y*w + x];
}
```

Figure 3: Color-to-grayscale and grayscale-to-3 channel kernels.

```
__global__ void conv2d(byte *d1, byte *d3,
        int h1, int w1, int h2, int w2)
{
        int y, x, i, j, imin, imax, jmin, jmax, ip, jp;
        float sum;

        // infer y, x, from block/thread index
        y = blockDim.y * blockIdx.y + threadIdx.y;
        x = blockDim.x * blockIdx.x + threadIdx.x;

        // out of bounds, no work to do
        if (x >= w1 || y >= h1) {
                return;
        }

        // appropriate ranges for convolution
        imin = max(0, y+h2/2-h2+1);
        imax = min(h1, y+h2/2+1);
        jmin = max(0, x+w2/2-w2+1);
        jmax = min(w1, x+w2/2+1);

        // convolution
        sum = 0;
        for (i = imin; i < imax; ++i) {
                for (j = jmin; j < jmax; ++j) {
                        ip = i - h2/2;
                        jp = j - w2/2;

                        sum += d1[i*w1 + j] * dFlt[(y-ip)*w2 + (x-jp)];
                }
        }

        // set result
        d3[y*w1 + x] = sum;
}
```

Figure 4: Naive convolution kernel

the kernel into constant memory, we get the (still naive) implementation found in Figure 4. This simple optimization halved the runtime of the convolution kernel, probably due to the fact that there were half as many global memory accesses.

The first optimization we can make is to make the convolution a "tiled" memory algorithm, in which we preload a section of the image into the block shared memory so that we reduce the number of global memory accesses. However, we have to be careful to load all of the pixels necessary for the convolution: the convolution for one block of pixels requires a surrounding "apron" of adjacent pixels, as shown in Figure 5a. This means that each thread loads one pixel, and some threads on the border of the thread block will be loading pixels but not performing a filter calculation; or, conversely, each thread performs a convolution operation but threads on the border of the thread block also load

adjacent apron pixels into shared memory. Either approach is fine, but we chose the former for its simplicity. This means that in each thread block, some threads are dedicated only to loading apron pixels. Now, each thread only performs one global memory access; however, we need more threads (more blocks) to process the image because not all threads are used anymore for calculating a convolution. Each pixel is loaded from global memory up to nine times, a huge improvement for larger kernels.

Note that having a large apron is wasteful; each apron pixel is loaded into shared memory several times and the corresponding threads are inactive during the filter computation. Thus, if the apron (filter) size is large relative to the block size, this becomes increasingly inefficient. In the example shown in Figure 5b, only one-ninth of the pixels actually perform filter calculations, and each pixel gets loaded from global memory exactly nine times. This is similar to the case of a 31×31 Gaussian filter in a 32×32 thread block.

The solution to our woes is the fact that a 2D Gaussian convolution is separable – we can decompose it into a vertical (1D) convolution followed by a horizontal (1D) convolution. The reason for the efficiency is two-fold: firstly, we only have to perform $2H$ rather than H^2 mult-adds, because our filter is linear (with the same length as the side-length of the 2D filter). Secondly, we only need to worry about loading apron pixels along the convolution axis. Each apron pixel gets loaded exactly twice from global memory. Furthermore, if we elongate the block so that its dimension along the convolution axis is longer than in the perpendicular direction, we can minimize the number of apron pixels loaded into memory; this is shown in Figure 5c.

These principles were applied to result in the 1-D convolution kernel shown in Figure 6. Only the horizontal filter is shown, but the same concept is applied to create a vertical filter, and the two are performed in serial. For the horizontal convolution, the block size is set to 16×64; for the vertical convolution, the block size is set to 64×16.

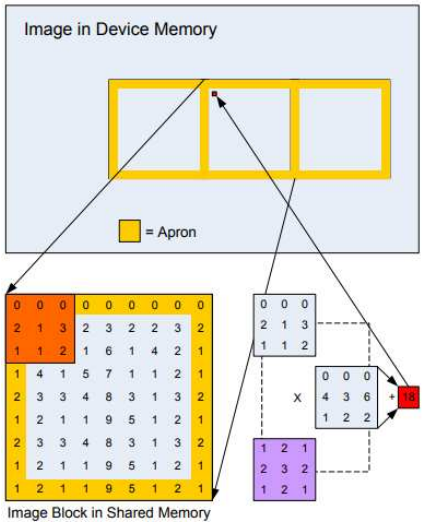
3.5 Sobel filter

The Sobel filter is simply a 2-D convolution, but it is a small fixed size (3×3). It is also separable into two linear filters (3×1 and 1×3), which reduces the number of computations. Given the small filter size and the few global memory accesses, we found that the optimizations did not help by much.

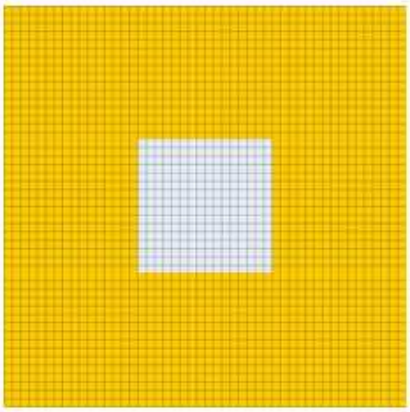
The naive version, shown in Figure 7, performed only slightly worse than the version using shared memory and separable filters, shown in Figure 8. A version using shared memory but without separating the filter, shown in Figure 9, performs slightly better than the separable version.

3.6 Non-maximum edge suppression and double-thresholding

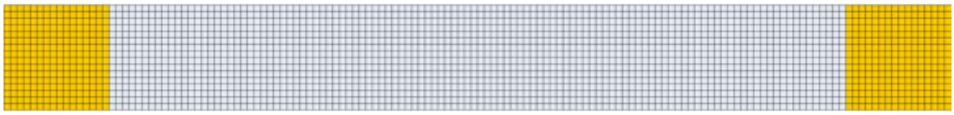
The implementation for these two sections are fairly straightforward and can't be optimized much. They each require few global memory accesses, do not



(a) Visual of the convolution apron



(b) An large (inefficient) apron



(c) Efficiency with a 1D convolution

Figure 5: Considerations with the convolution apron. Images source: [3].

```
__global__ void conv1dRows(byte *dIn, byte *dOut, int h, int w, int fltSize)
{
        int y, x, as, i, j;
        float sum;
        __shared__ byte tmp[lbs*sbs];

        as = fltSize>>1; // apron size

        // infer y, x, from block/thread index
        // note extra operations based on apron for x
        y = sbs * blockIdx.y + ty;
        x = (lbs-(as<<1)) * blockIdx.x + tx-as;

        // load data
        if (y<h && x>=0 && x<w) {
                tmp[ty*lbs+tx] = dIn[y*w+x];
        }

        __syncthreads();

        // perform 1-D convolution
        if (tx>=as && tx<lbs-as && y<h && x<w) {
                for (i = ty*lbs+tx-as, j = 0, sum = 0; j < fltSize; ++i, ++j) {
                        sum += dFlt[j] * tmp[i];
                }

                // set result
                dOut[y*w+x] = sum;
        }
}
```

Figure 6: Efficient 1-D convolution kernel

```
__global__ void sobel(byte *img, byte *out, byte *out2, int h, int w)
{
        int vKer, hKer, y, x;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        // make sure not on edge
        if (y <= 0 || y >= h-1 || x <= 0 || x >= w-1) {
                return;
        }

        vKer = img[(y-1)*w+(x-1)]*1 + img[(y-1)*w+x]*2 + img[(y-1)*w+(x+1)]*1 +
                img[(y+1)*w+(x-1)]*-1 + img[(y+1)*w+x]*-2 + img[(y+1)*w+(x+1)]*-1;

        hKer = img[(y-1)*w+(x-1)]*1 + img[(y-1)*w+(x+1)]*-1 +
                img[y*w+(x-1)]*2 + img[y*w+(x+1)]*-2 +
                img[(y+1)*w+(x-1)]*1 + img[(y+1)*w+(x+1)]*-1;

        out[y*w+x] = out[y*w+x] = sqrtf(hKer*hKer + vKer*vKer);
        out2[y*w+x] = (byte)((atan2f(vKer,hKer)+9/8*M_PI)*4/M_PI)&0x3;
}
```

Figure 7: Naïve Sobel convolution

cause any major branch diversions, and as a result take the least time of all the kernels. See Figures 10 and 11 for the implementations.

3.7 Edge tracking via hysteresis

This method closely mirrors that of Luo and Duraiswami [1]. The general idea is to emulate a BFS using tiled memory. What makes it difficult, as Luo and Duraiswami point out, is that this is a nonlocal problem; unlike any of the other kernels so far, one "strong" edge may affect a "weak" edge at an arbitrary distance connected by some path. We follow the heuristic multi-pass approach given by Luo and Duraiswami. The general idea is to introduce an apron of width 1 so that a block can "discover" strong edges from adjacent blocks. In each pass, the block performs a local iterative BFS, updating weak edges connected to strong edges until there are no more remaining. After each pass, the results are written back to global memory so they can be retrieved by adjacent blocks in the next kernel invocation. A visual of the algorithm is shown in Figure 12.

Like Luo and Duraiswami, we choose an arbitrary number of hysteresis passes to perform (we perform five iterations). The effect of increasing the number of passes is shown in Figure 13.

The abundant `__syncthreads()` usage is probably not optimal, but it was a necessary workaround to prevent race conditions.

The initial implementation using global memory is shown in Figure 14. The

```
__global__ void sobel_sep(byte *img, byte *out, byte *out2, int h, int w)
{
        int y, x;

        // using int instead of byte for the following offers a 0.01s (5%)
        // speedup on the 16k image -- coalesced memory?
        int vKer, hKer;
        __shared__ int tmp1[bs*bs], tmp2[bs*bs], tmp3[bs*bs];

        y = (bs-2)*blockIdx.y + threadIdx.y-1;
        x = (bs-2)*blockIdx.x + threadIdx.x-1;

        // load data from image
        if (y>=0 && y<h && x>=0 && x<w) {
                tmp1[ty*bs+tx] = img[y*w+x];
        }

        __syncthreads();

        // first convolution
        if (ty>=1 && ty<bs-1 && tx && tx<bs) {
                tmp2[ty*bs+tx] = tmp1[(ty-1)*bs+tx]
                        + (tmp1[ty*bs+tx]<<1) + tmp1[(ty+1)*bs+tx];
        }

        if (ty && ty<bs && tx>=1 && tx<bs-1) {
                tmp3[ty*bs+tx] = tmp1[ty*bs+(tx-1)]
                        + (tmp1[ty*bs+tx]<<1) + tmp1[ty*bs+(tx+1)];
        }

        __syncthreads();

        // second convolution and write-back
        if (ty>=1 && ty<bs-1 && tx>=1 && tx<bs-1 && y<h && x<w) {
                hKer = tmp2[ty*bs+(tx-1)] - tmp2[ty*bs+(tx+1)];
                vKer = tmp3[(ty-1)*bs+tx] - tmp3[(ty+1)*bs+tx];

                out[y*w+x] = sqrtf(hKer*hKer + vKer*vKer);
                out2[y*w+x] = (byte)((atan2f(vKer,hKer)+9/8*M_PI)*4/M_PI)&0x3;
        }
}
```

Figure 8: Sobel convolution using shared memory and separable filters

```
__global__ void sobel_shm(byte *img, byte *out, byte *out2, int h, int w)
{
        int y, x;
        int vKer, hKer;
        __shared__ int tmp[bs*bs];

        y = (bs-2)*blockIdx.y + threadIdx.y-1;
        x = (bs-2)*blockIdx.x + threadIdx.x-1;

        // load data from image
        if (y>=0 && y<h && x>=0 && x<w) {
                tmp[ty*bs+tx] = img[y*w+x];
        }

        __syncthreads();

        // convolution and write-back
        if (ty>=1 && ty<bs-1 && tx>=1 && tx<bs-1 && y<h && x<w) {
                vKer = tmp[(ty-1)*bs+(tx-1)]*1 + tmp[(ty-1)*bs+tx]*2
                        + tmp[(ty-1)*bs+(tx+1)]*1 + tmp[(ty+1)*bs+(tx-1)]*-1
                        + tmp[(ty+1)*bs+tx]*-2 + tmp[(ty+1)*bs+(tx+1)]*-1;

                hKer = tmp[(ty-1)*bs+(tx-1)]*1 + tmp[(ty-1)*bs+(tx+1)]*-1 +
                       tmp[ty*bs+(tx-1)]*2 + tmp[ty*bs+(tx+1)]*-2 +
                       tmp[(ty+1)*bs+(tx-1)]*1 + tmp[(ty+1)*bs+(tx+1)]*-1;

                out[y*w+x] = sqrtf(hKer*hKer + vKer*vKer);
                out2[y*w+x] = (byte)((atan2f(vKer,hKer)+9/8*M_PI)*4/M_PI)&0x3;
        }
}
```

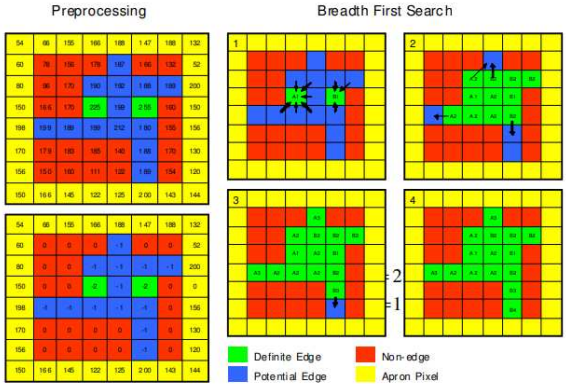
Figure 9: Sobel convolution using shared memory and 2D filter

```
__global__ void edge_thin(byte *mag, byte *angle, byte *out, int h, int w)
{
        int y, x, y1, x1, y2, x2;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        // make sure not on the border
        if (y <= 0 || y >= h-1 || x <= 0 || x >= w-1) {
                return;
        }

        // if not greater than angles in both directions, then zero
        switch (angle[y*w + x]) {
        case 0:
                // horizontal
                y1 = y2 = y;
                x1 = x-1;
                x2 = x+1;
                break;
        case 3:
                // 135
                y1 = y-1;
                x1 = x+1;
                y2 = y+1;
                x2 = x-1;
                break;
        case 2:
                // vertical
                x1 = x2 = x;
                y1 = y-1;
                y2 = y+1;
                break;
        case 1:
                // 45
                y1 = y-1;
                x1 = x-1;
                y2 = y+1;
                x2 = x+1;
        }

        if (mag[y1*w + x1] >= mag[y*w + x] || mag[y2*w + x2] >= mag[y*w + x]) {
                out[y*w + x] = 0;
        } else {
                out[y*w + x] = mag[y*w + x];
        }
}
```

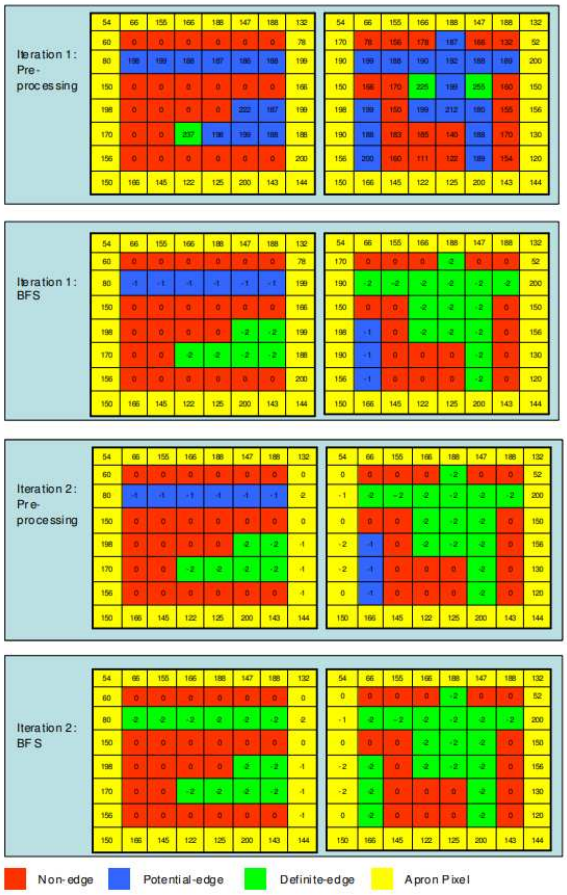
Figure 10: Edge thinning kernel

```
#define MSK_LOW         0x0     // below threshold 1
#define MSK_THR         0x60    // at threshold 1
#define MSK_NEW         0x90    // at threshold 2, newly discovered
#define MSK_DEF         0xff    // at threshold 2 and already discovered

// perform double thresholding
__global__ void edge_thin(byte *dImg, byte *out, int h, int w, byte t1, byte t2)
{
        int y, x, ind, grad;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        if (y >= h || x >= w) {
                return;
        }

        ind = y*w + x;
        grad = dImg[ind];
        if (grad < t1) {
                out[ind] = MSK_LOW;
        } else if (grad < t2) {
                out[ind] = MSK_THR;
        } else {
                out[ind] = MSK_NEW;
        }
}
```

Figure 11: Double-thresholding kernel

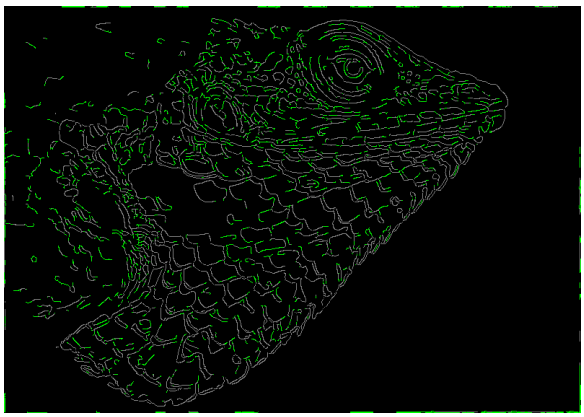


(a) Hysteresis edge tracking within a block (BFS in shared memory)

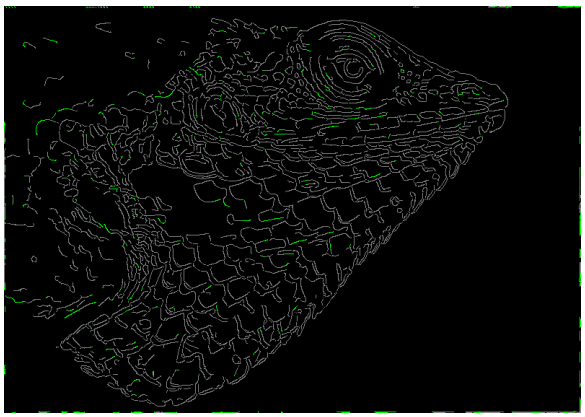


(b) Edge tracking across blocks (multiple passes)

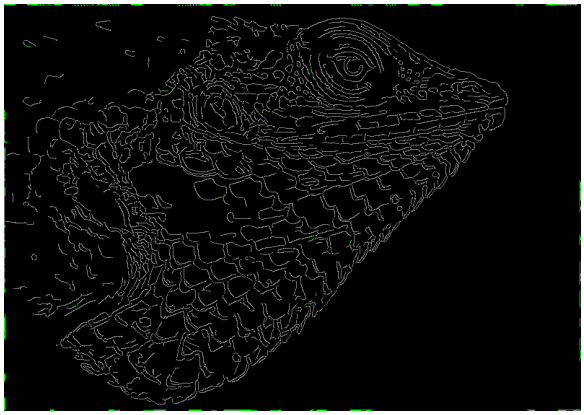
Figure 12: Hysteresis algorithm. Images source: [1].



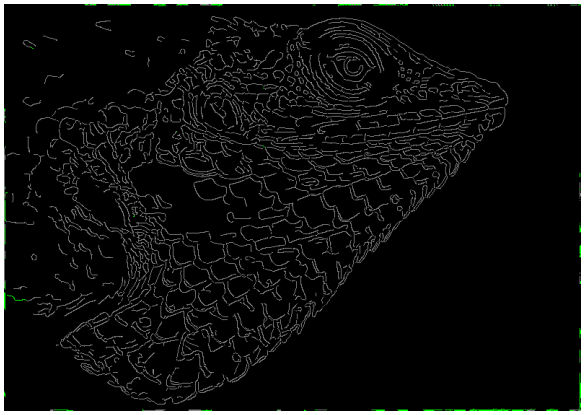
(a) 1 pass



(b) 2 passes



(c) 3 passes



(d) 4 passes

Figure 13: Effect of varying the number of hysteresis passes on the lizard image. The green pixels indicate new strong edge pixels found in the current hysteresis pass, and gray pixels indicate edges that were already marked strong before the current pass. Very few edges are added after the third iteration. We chose five iterations as the default for our tests.

tiled version is shown in Figure 15.

3.8 CPU implementation

The CPU implementation follows the same logic as the CUDA naive versions. We cannot make some of the optimizations (e.g., shared memory – the CPU cache is not large enough). Most of the algorithms were implemented on a pixel-by-pixel basis by simply looping through every pixel with a double loop.

The main differences lie in the setup and the hysteresis algorithm. The CPU version does not require copying the image buffer from host to device.

The hysteresis version is not performed in parallel passes like the CUDA version for obvious reasons. Instead, a DFS is performed on each pixel. Since this involves nonlocal memory that requires expensive synchronization between blocks (i.e., writing to) on the CUDA version – this makes the speedup on the hysteresis section the smallest of all of the kernels.

The only optimization that was performed on the GPU that could also be performed on the CPU is the separable blurring filter, but that was not implemented. This is left for future investigation.

4 Results

For testing, we mostly focused on four large images downloaded from the image, which will henceforth be referred to as "lizard"[6], "rocks"[7], "moon"[8], and "skyline14k"[9]. Several scaled versions of the skyline image were also used.

As expected, the grayscale, edge thinning, and thresholding operators are the quickest. These all involve a single linear pass over the data and only a few memory accesses per pixel. The sobel and blurring operators take the most time. These results can be seen in Tables 1 and 2. The mean time reduction for the edge thinning and thresholding from the CPU to CUDA versions is 93% and 86%, respectively.

The hysteresis operator is one of the faster operators for the CPU version, while it is the slowest operator (counting overall time for five iterations of the kernel) on the CUDA version. This is as expected: on the CPU, we have much faster control flow operators and nonlocal caching using a DFS, However, in CUDA, since each warp has to perform the same instruction, when any path is being traced all of the other threads in the warp must remain idle until all paths are traced. Additionally, we need multiple global memory write-backs because edges may pass between thread block tile boundaries. As a result, the CPU hysteresis is actually faster than the CUDA hysteresis for the smallest test image (lizard), and on the same order of magnitude for the other three test images.

[6] Babujayan, CC BY 3.0, via Wikimedia Commons.
[7] `https://wallpapercave.com/w/wp1848524`
[8] `https://www.reddit.com/r/pics/comments/dszr27`
[9] Kristoffer Trolle from Copenhagen, Denmark, CC BY 2.0, via Wikimedia Commons.

```
// check and set neighbor
#define CAS(buf, cond, x2, y2, width) \
        if ((cond) && (buf)[(y2)*(width)+(x2)] == MSK_THR) \
                (buf)[(y2)*(width)+(x2)] = MSK_NEW

// perform one iteration of hysteresis
__global__ void hysteresis(byte *dImg, int h, int w, bool final)
{
        int y, x;
        __shared__ byte changes;

        // infer y, x, from block/thread index
        y = blockDim.y * blockIdx.y + threadIdx.y;
        x = blockDim.x * blockIdx.x + threadIdx.x;

        // check if pixel is connected to its neighbors; continue until
        // no changes remaining
        do {
                __syncthreads();
                changes = 0;
                __syncthreads();

                // make sure inside bounds -- need this here b/c we can't have
                // __syncthreads() cause a branch divergence in a warp;
                // see https://stackoverflow.com/a/6667067/2397327

                // if newly-discovered edge, then check its neighbors
                if ((x<w && y<h) && dImg[y*w+x] == MSK_NEW) {
                        // promote to definitely discovered
                        dImg[y*w+x] = MSK_DEF;
                        changes = 1;

                        // check neighbors
                        CAS(dImg, x>0&&y>0,     x-1, y-1, w);
                        CAS(dImg, y>0,          x,   y-1, w);
                        CAS(dImg, x<w-1&&y>0,   x+1, y-1, w);
                        CAS(dImg, x<w-1,        x+1, y,   w);
                        CAS(dImg, x<w-1&&y<h-1, x+1, y+1, w);
                        CAS(dImg, y<h-1,        x,   y+1, w);
                        CAS(dImg, x>0&&y<h-1,   x-1, y+1, w);
                        CAS(dImg, x>0,          x-1, y,   w);
                }

                __syncthreads();
        } while (changes);

        // set all threshold1 values to 0
        if (final && (x<w && y<h) && dImg[y*w+x] != MSK_DEF) {
                dImg[y*w+x] = 0;
        }
}
```

Figure 14: Naïve hysteresis using global memory

```
__global__ void hysteresis_shm(byte *dImg, int h, int w, bool final)
{
        int y, x;
        bool in_bounds;
        __shared__ byte changes, tmp[bs*bs];

        // infer y, x, from block/thread index
        y = (bs-2)*blockIdx.y + ty-1;
        x = (bs-2)*blockIdx.x + tx-1;

        in_bounds = (x<w && y<h) && (tx>=1 && tx<bs-1 && ty>=1 && ty<bs-1);

        if (y>=0 && y<h && x>=0 && x<w) {
                tmp[ty*bs+tx] = dImg[y*w+x];
        }

        __syncthreads();

        // check if pixel is connected to its neighbors; continue until
        // no changes remaining
        do {
                __syncthreads();
                changes = 0;
                __syncthreads();

                // if newly-discovered edge, then check its neighbors
                if (in_bounds && tmp[ty*bs+tx] == MSK_NEW) {
                        // promote to definitely discovered
                        tmp[ty*bs+tx] = MSK_DEF;
                        changes = 1;

                        // check neighbors
                        CAS(tmp, 1,             tx-1, ty-1, bs);
                        CAS(tmp, 1,             tx,   ty-1, bs);
                        CAS(tmp, x<w-1,         tx+1, ty-1, bs);
                        CAS(tmp, x<w-1,         tx+1, ty,   bs);
                        CAS(tmp, x<w-1&&y<h-1, tx+1, ty+1, bs);
                        CAS(tmp, y<h-1,         tx,   ty+1, bs);
                        CAS(tmp, y<h-1,         tx-1, ty+1, bs);
                        CAS(tmp, 1,             tx-1, ty,   bs);
                }

                __syncthreads();
        } while (changes);

        if (y>=0 && y<h && x>=0 && x<w) {
                if (final) {
                        if (in_bounds) {
                                dImg[y*w+x] = MSK_DEF*(tmp[ty*bs+tx]==MSK_DEF);
                        }
                } else {
                        dImg[y*w+x] = max(dImg[y*w+x], tmp[ty*bs+tx]);
                }
        }
}
```

Figure 15: Hysteresis using shared memory

We were able to achieve minor speedups with each of the optimizations. The previously discussed results are the most optimized versions. Table 3 lists the times of the less-optimized versions (except for the unoptimized blur kernel, which will be discussed separately.) Switching the grayscale operator from using floating-point operations to integral ones caused a 26% decrease in kernel time. Switching hysteresis to use shared memory caused a 52% speedup (this is displayed visually in Figure 16a).

The speedup from the Sobel filter optimizations was milder, likely due to the fact that it is already a relatively simple operator. Using shared memory (tiling) with a separable filter is actually a little slower than using a non-separable filter, most likely because of requiring two thread barriers rather than a single one. The separable version offers a 7% time reduction, and the non-separable version offers a 10% reduction over the naive implementation. See Figure 16b.

Lastly, an additional "optimization" to remove all `cudaDeviceSynchronize()` calls was made to see the effect of these synchronization barriers. There is a consistent but very small effect (on the order of thousandths of seconds for each test image).

In Tables 4 and 5, we see that the times scale very linearly with the number of pixels in the image. (This is true for all but the CPU blur filter, which will be discussed in the following paragraph.) This is shown visually in Figure 16d.

The largest speedup by far was achieved in the blur filter, not least because it involves the most computations and is the slowest kernel in general. Table 6 and Figure 16c demonstrates the quadratic growth of the non-separable version as the blur size increases, as opposed to the linear growth with the separable version. For most of our tests, the blur size had $\sigma = 2$, at which the time spent is reduced on average by 82%. For $\sigma = 5$, the reduction is 93%. (The (normalized) time reduction is unbelievably consistent, varying by less than 1% – what appears to be two lines in Figure 16c is actually 8 lines, with each of the four images represented.)

The CPU version implements the naive blur and thus is also clearly quadratic. For $\sigma = 2$ ($H = 13$, which means 169 memory accesses and mult-adds per pixel), the naive CUDA version (using the same algorithm) offers a 99.3% time reduction, and thus the optimized CUDA version offers a 99.98% speedup. Since the overall time (for the CPU version) is dominated by the blurring time, the overall time reduction for the images is also over 99%.

In Table 7, the images produced by the CPU and CUDA versions are compared. The resulting accuracy metric is simply the percentage of pixels that match in the final image. Some of the smaller images report a larger error margin; this is probably due to the fact that the boundary pixels are not properly accounted for in either of the latest implementations during the convolutions. This should be fixed for a future comparison. Much of the remaining differences should result from hysteresis, as this is the only non-deterministic algorithm (in the CUDA version only, as it depends on which warps get scheduled to run first). However, the numbers are generally high enough to indicate that most pixels were the same; the mean accuracy over the test images is roughly 99%.

image	lizard	rocks	moon	skyline14k
width	4444	7680	14694	14091
height	3136	4320	8266	9394
pixels (MP)	13.94	33.1776	121.46	132.37
blur	16.392047	38.991000	144.943194	158.283468
sobel	0.503448	1.370735	4.346238	5.109478
edgethin	0.127931	0.392264	1.259965	1.390145
threshold	0.044430	0.096321	0.345455	0.407205
hysteresis	0.039386	0.154966	0.443695	0.559849
overall	17.294973	41.319512	152.664951	167.053442

Table 1: CPU timings for processing the same images as in Table 2.

image	lizard	rocks	moon	skyline14k
gray	0.007841	0.017504	0.068982	0.073921
blur (separable)	0.024333	0.057933	0.207798	0.225898
sobel (shared mem)	0.019585	0.052028	0.201424	0.203151
edgethin	0.009991	0.024648	0.090547	0.098592
threshold	0.005778	0.013698	0.049114	0.054493
hysteresis (shared mem)	0.008176	0.017076	0.058182	0.068109
hyst total	0.040881	0.085383	0.290910	0.340547
overall	0.117525	0.270621	0.983041	1.076650
nosync	0.116558	0.267512	0.979548	1.071600

Table 2: GPU kernel and overall timings for optimized kernels on four test images. For these tests, the following hyperparameters were used: blur $\sigma = 2$, threshold 1=0.2, threshold 2=0.4, and 5 hysteresis passes. Additionally, the nosync row indicates the overall time if no `cudaDeviceSynchronize()` calls were made; the effect of `cudaDeviceSynchronize()` is fairly negligible.

image	lizard	rocks	moon	skyline14k
gray (fp)	0.010738	0.023893	0.092579	0.100435
sobel (naive)	0.022106	0.056393	0.225306	0.222554
sobel (separable)	0.020585	0.052922	0.206515	0.210145
hysteresis (naive)	0.016735	0.034941	0.120156	0.139790
hyst total	0.083675	0.174705	0.600781	0.698950

Table 3: GPU kernel timings for unoptimized kernels on the same four test images. The same canny parameters were used.

image	skyline500	skyline1000	skyline2.5k	skyline5k	skyline10k	skyline14k
width	500	1000	2500	5000	10000	14091
height	333	667	1667	3334	6667	9394
pixels (MP)	0.17	0.67	4.17	16.67	66.67	132.37
blur	0.193700	0.779473	4.867015	19.478580	79.527461	158.283468
sobel	0.006869	0.027894	0.149842	0.584700	2.346041	5.109478
edgethin	0.001850	0.007012	0.039814	0.154795	0.613782	1.390145
threshold	0.000578	0.002514	0.012705	0.048452	0.189971	0.407205
hysteresis	0.001592	0.006575	0.022750	0.079599	0.293580	0.559849
overall	0.206662	0.830951	5.131639	20.503200	83.607380	167.053442

Table 4: CPU timings for processing the same images as in Table 5.

image	skyline500	skyline1k	skyline2.5k	skyline5k	skyline10k	skyline14k
gray	0.000265	0.000737	0.002767	0.009557	0.035608	0.073921
blur	0.000756	0.001451	0.007824	0.029554	0.113994	0.225898
sobel	0.000299	0.001084	0.006579	0.025391	0.101429	0.203151
edgethin	0.000385	0.000772	0.003750	0.012856	0.049072	0.098592
threshold	0.000321	0.000521	0.001737	0.007436	0.027187	0.054493
hysteresis	0.000243	0.000551	0.002589	0.009209	0.034696	0.068109
hysteresis total	0.001217	0.002757	0.012947	0.046046	0.173481	0.340547
overall	0.003606	0.008689	0.039141	0.141744	0.539666	1.076650

Table 5: GPU kernel timings for optimized kernels on the same image at six resolutions.

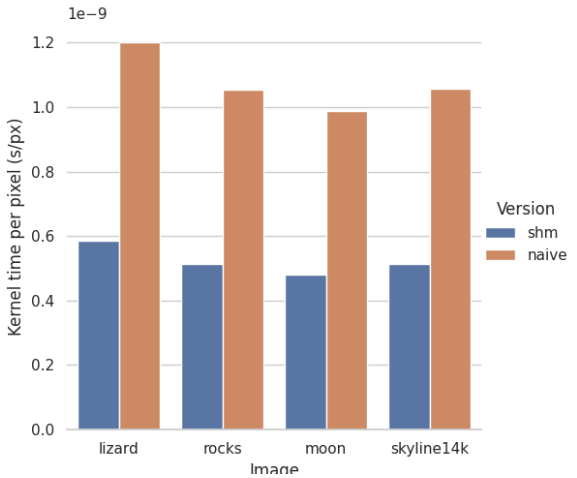
image	lizard	rocks	moon	skyline14k
naive $\sigma = 1$	0.048326	0.115943	0.421206	0.456401
naive $\sigma = 2$	0.132040	0.313778	1.153320	1.254950
naive $\sigma = 3$	0.277972	0.658977	2.413430	2.624620
naive $\sigma = 4$	0.473423	1.119180	4.125480	4.493010
naive $\sigma = 4$	0.707206	1.691080	6.175220	6.761020
separable $\sigma = 1$	0.018326	0.042776	0.157439	0.169571
separable $\sigma = 2$	0.024333	0.057933	0.207798	0.225898
separable $\sigma = 3$	0.029375	0.068700	0.249971	0.274145
separable $\sigma = 4$	0.037768	0.088535	0.325202	0.355358
separable $\sigma = 5$	0.049688	0.117644	0.426353	0.465503

Table 6: Comparison of timings for the naive and optimized Gaussian convolutions on the four test images for different blur standard deviations.

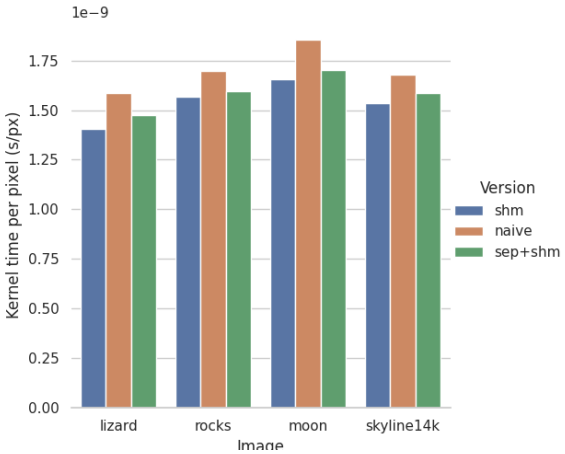
image	lizard	moon	rocks	skyline14k
accuracy	0.9767	0.994778	0.988704	0.994285

image	skyline500	skyline1k	skyline2.5k	skyline5k	skyline10k
accuracy	0.968613	0.978121	0.985476	0.990231	0.992143

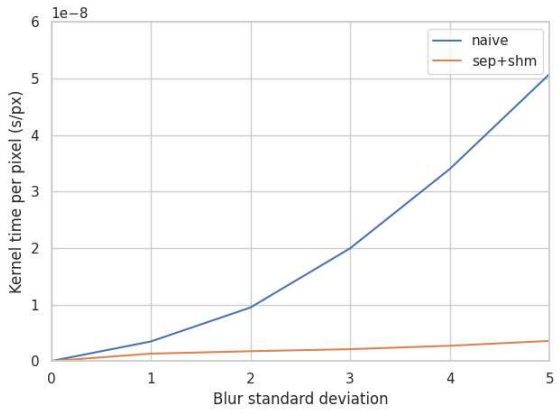
Table 7: Percent of pixels matching from CPU and GPU versions



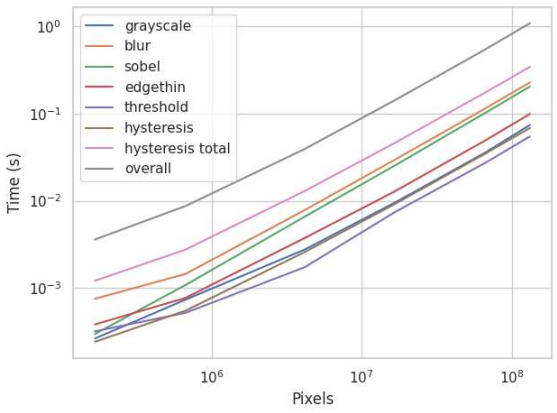
(a) Comparison of normalized kernel times of the two hysteresis implementations.



(b) Comparison of normalized kernel times of the three Sobel filter implementations. The two optimized version perform very similarly, but the non-separable one probably wins out due to having fewer synchronization barriers.



(c) Comparison of the normalized kernel times of the two (Gaussian) convolution implementations, shown over several standard deviations $\sigma = \{0, 1, 2, 3, 4, 5\}$. Not only does the separable filter beat the 2D filter by a large margin, but it scales linearly with blur size rather than quadratically.



(d) Timings of the different kernels on images of different sizes. All of the kernel timings grow linearly w.r.t. the number of pixels in the image, as expected. This is performed on six different resolutions of the cityscape image with the same Canny parameters.

Figure 16: Result of optimizing kernels, and overall scalability. (a), (b), and (c) are normalized by the number of pixels in the image. (d) is shown on a log-log scale and demonstrates that all of the algorithms scale linearly with number of pixels (this allows us to normalize by pixels count and get similar results).

5 Conclusion

We were able to achieve a 99.4% speedup using optimized CUDA C++ over a naive implementation on the CPU written in C on four large test images. Simply by converting what would a pixel-by-pixel algorithm to a parallel kernel offered over 90% time reduction for all kernels except hysteresis. Most of the time reduction came from the blurring kernel; the naive version achieved a 99.3% time reduction and the optimized version caused a 99.98% time reduction. As expected, the hysteresis kernel runtime is comparable to that of the CPU version, because of the nonlocal memory accesses and complex nonuniform branching requirements.

For future research, we may attempt separating the blur kernel for the CPU in the same manner as for CUDA. Further optimizations for the CUDA version may include optimizing for the capabilities of the device (our target was the GT 740, but its capabilities may be different from newer NVIDIA processors) and accounting for memory coalescence by properly aligning tiles. It may also be a good idea to test this against comparable versions written in OpenCV/OpenCL and MATLAB; Luo and Duraiswami [1] perform tests against OpenCV and MATLAB, but the hardware capabilities and software improvements (e.g., the introduction of OpenCL) since then have greatly changed.

References

[1] Yuancheng Luo and Ramani Duraiswami. Canny edge detection on nvidia cuda. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.

[2] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[3] Victor Podlozhnyuk. Image convolution with cuda. *NVIDIA Corporation white paper, June*, 2097(3), 2007.

Canny Filter Implementation in CUDA C++

ECE453 Final Project

Jonathan Lam

The Cooper Union for the Advancement of Science and Art

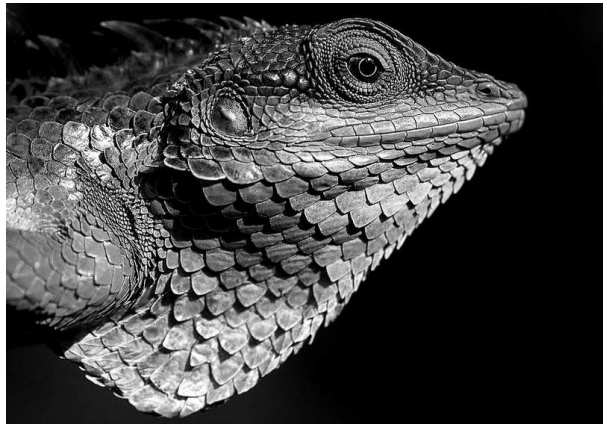
May 10, 2021

Objective:

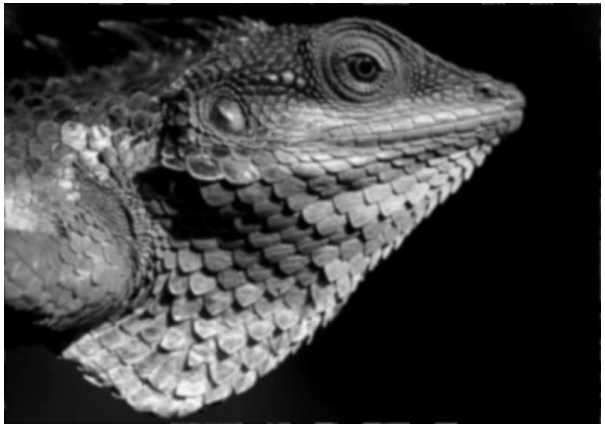
- Implement Canny edge detection algorithm in CUDA C++ and CPU (C/C++)
- Compare performance differences and accuracy
- Attempt further optimizations in CUDA

Work is based on Luo and Duraiswami [2].

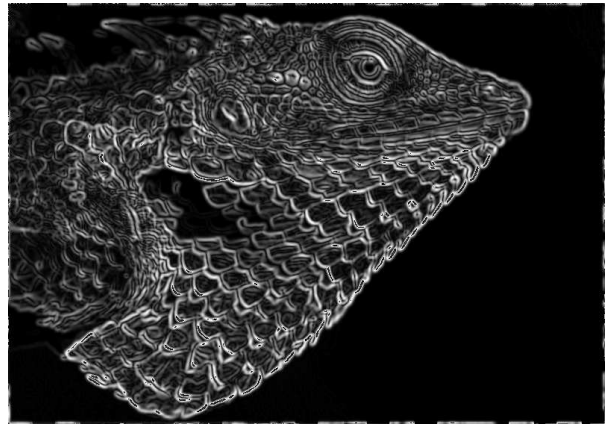
(a) Original image



(b) Luminance operator

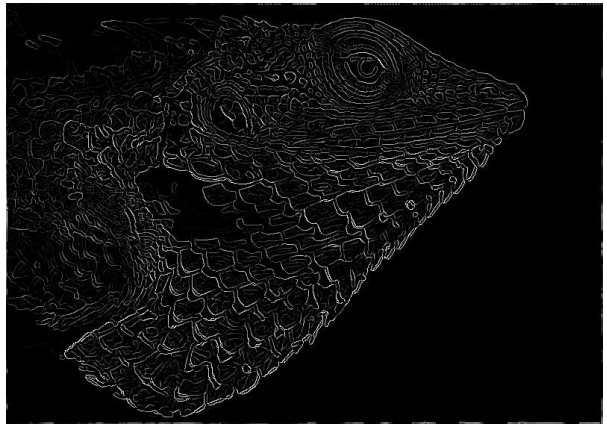


(c) Gaussian blur

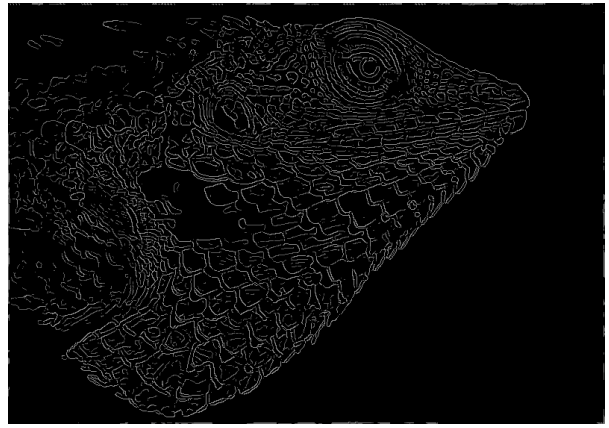


(d) Sobel filter

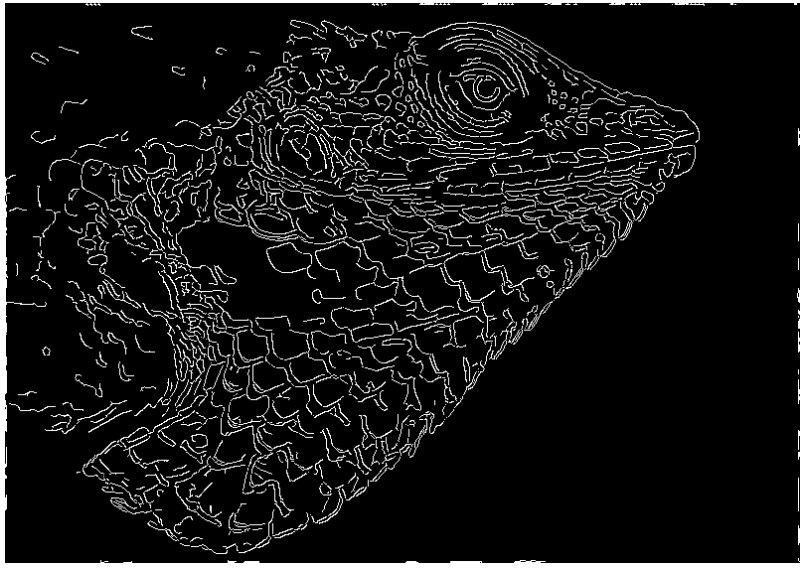
Figure: Canny edge detection stages. Zoom in for detail. Babujayan, CC BY 3.0, via Wikimedia Commons.



(e) Edge thinning

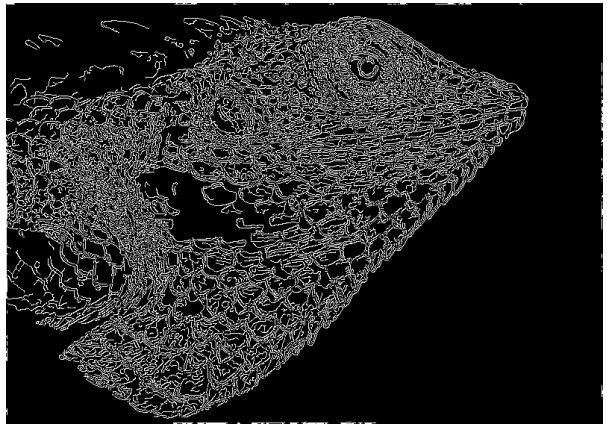


(f) Double thresholding

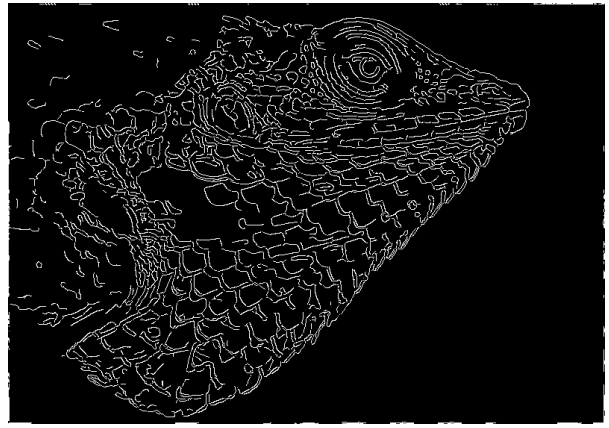


(g) Finished canny

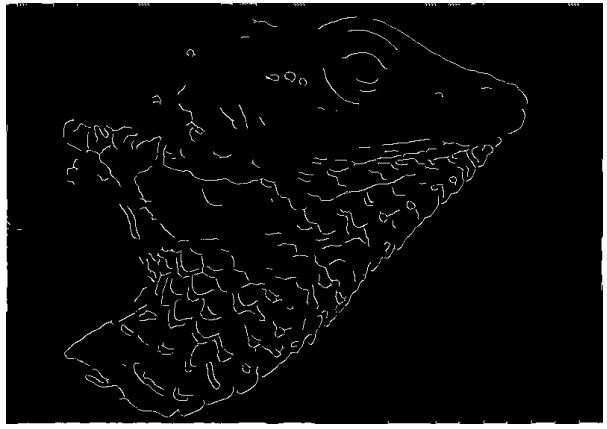
Figure: Canny edge detection stages. Zoom in for detail. Babujayan, CC BY 3.0, via Wikimedia Commons.



(a) $\sigma = 1$



(b) $\sigma = 2$



(c) $\sigma = 3$

(d) $\sigma = 4$

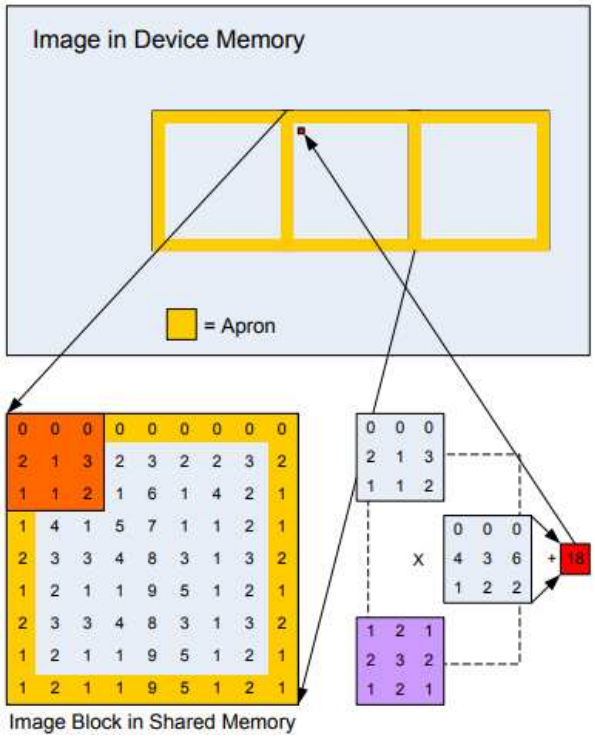
Figure: (cont'd.) Effect of changing blur standard deviation on the same lizard image. The blur size is dependent on the detail density (resolution) of the image. Increasing the blur size will decrease noise as well as the number of detected edges. It may also be necessary to change the thresholds based on the blur size. $\sigma = 2$ was chosen as the default for our tests as that seemed to work fairly well for reasonable resolutions.

```
__global__ void toGrayScale(byte *dImg, byte *dImgMono, int h, int w, int ch)
{
        int ind, y, x;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        if (y >= h || x >= w) {
                return;
        }

        ind = y*w*ch + x*ch;

        dImgMono[y*w + x] = (2989*dImg[ind] + 5870*dImg[ind+1]
                + 1140*dImg[ind+2])/10000;
}

// convert back from single channel to multi-channel
__global__ void fromGrayScale(byte *dImgMono, byte *dImg, int h, int w, int ch)
{
        int ind, y, x;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        if (y >= h || x >= w) {
                return;
        }

        ind = y*w*ch + x*ch;
        dImg[ind] = dImg[ind+1] = dImg[ind+2] = dImgMono[y*w + x];
}
```

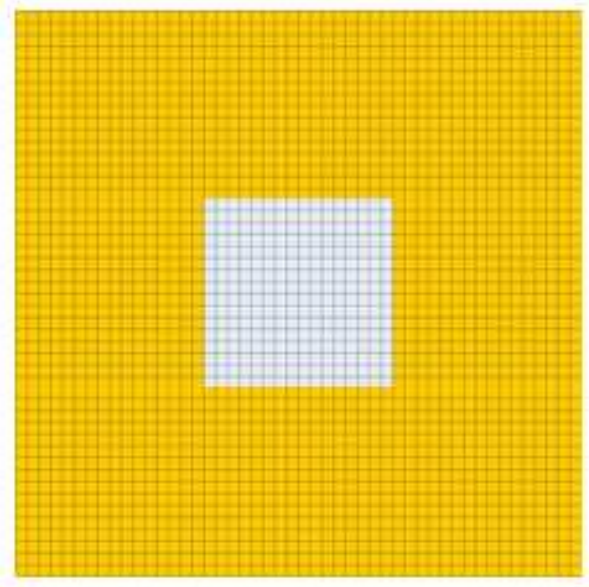
Figure: Color-to-grayscale and grayscale-to-3 channel kernels.

```
__global__ void conv2d(byte *d1, byte *d3,
        int h1, int w1, int h2, int w2)
{
        int y, x, i, j, imin, imax, jmin, jmax, ip, jp;
        float sum;

        // infer y, x, from block/thread index
        y = blockDim.y * blockIdx.y + threadIdx.y;
        x = blockDim.x * blockIdx.x + threadIdx.x;

        // out of bounds, no work to do
        if (x >= w1 || y >= h1) {
                return;
        }

        // appropriate ranges for convolution
        imin = max(0, y+h2/2-h2+1);
        imax = min(h1, y+h2/2+1);
        jmin = max(0, x+w2/2-w2+1);
        jmax = min(w1, x+w2/2+1);

        // convolution
        sum = 0;
        for (i = imin; i < imax; ++i) {
                for (j = jmin; j < jmax; ++j) {
                        ip = i - h2/2;
                        jp = j - w2/2;

                        sum += d1[i*w1 + j] * dFlt[(y-ip)*w2 + (x-jp)];
                }
        }

        // set result
        d3[y*w1 + x] = sum;
}
```

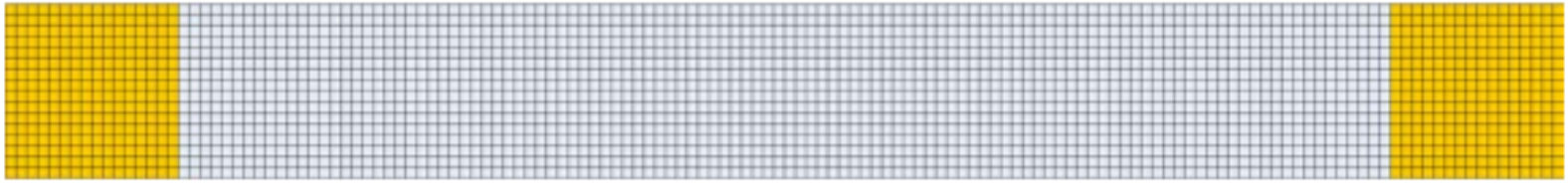
Figure: Naive convolution kernel



(a) Visual of the convolution apron



(b) An large (inefficient) apron



(c) Efficiency with a 1D convolution

Figure: Considerations with the convolution apron. Images source: [1].

```
__global__ void conv1dRows(byte *dIn, byte *dOut, int h, int w, int fltSize)
{
        int y, x, as, i, j;
        float sum;
        __shared__ byte tmp[lbs*sbs];

        as = fltSize>>1; // apron size

        // infer y, x, from block/thread index
        // note extra operations based on apron for x
        y = sbs * blockIdx.y + ty;
        x = (lbs-(as<<1)) * blockIdx.x + tx-as;

        // load data
        if (y<h && x>=0 && x<w) {
                tmp[ty*lbs+tx] = dIn[y*w+x];
        }

        __syncthreads();

        // perform 1-D convolution
        if (tx>=as && tx<lbs-as && y<h && x<w) {
                for (i = ty*lbs+tx-as, j = 0, sum = 0; j < fltSize; ++i, ++j) {
                        sum += dFlt[j] * tmp[i];
                }

                // set result
                dOut[y*w+x] = sum;
        }
}
```

Figure: Efficient 1-D convolution kernel

```
__global__ void sobel(byte *img, byte *out, byte *out2, int h, int w)
{
        int vKer, hKer, y, x;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        // make sure not on edge
        if (y <= 0 || y >= h-1 || x <= 0 || x >= w-1) {
                return;
        }

        vKer = img[(y-1)*w+(x-1)]*1 + img[(y-1)*w+x]*2 + img[(y-1)*w+(x+1)]*1 +
                img[(y+1)*w+(x-1)]*-1 + img[(y+1)*w+x]*-2 + img[(y+1)*w+(x+1)]*-1;

        hKer = img[(y-1)*w+(x-1)]*1 + img[(y-1)*w+(x+1)]*-1 +
                img[y*w+(x-1)]*2 + img[y*w+(x+1)]*-2 +
                img[(y+1)*w+(x-1)]*1 + img[(y+1)*w+(x+1)]*-1;

        out[y*w+x] = out[y*w+x] = sqrtf(hKer*hKer + vKer*vKer);
        out2[y*w+x] = (byte)((atan2f(vKer,hKer)+9/8*M_PI)*4/M_PI)&0x3;
}
```

Figure: Naïve Sobel convolution

```
__global__ void sobel_sep(byte *img, byte *out, byte *out2, int h, int w)
{
        int y, x;

        // using int instead of byte for the following offers a 0.01s (5%)
        // speedup on the 16k image -- coalesced memory?
        int vKer, hKer;
        __shared__ int tmp1[bs*bs], tmp2[bs*bs], tmp3[bs*bs];

        y = (bs-2)*blockIdx.y + threadIdx.y-1;
        x = (bs-2)*blockIdx.x + threadIdx.x-1;

        // load data from image
        if (y>=0 && y<h && x>=0 && x<w) {
                tmp1[ty*bs+tx] = img[y*w+x];
        }

        __syncthreads();

        // first convolution
        if (ty>=1 && ty<bs-1 && tx && tx<bs) {
                tmp2[ty*bs+tx] = tmp1[(ty-1)*bs+tx]
                        + (tmp1[ty*bs+tx]<<1) + tmp1[(ty+1)*bs+tx];
        }

        if (ty && ty<bs && tx>=1 && tx<bs-1) {
                tmp3[ty*bs+tx] = tmp1[ty*bs+(tx-1)]
                        + (tmp1[ty*bs+tx]<<1) + tmp1[ty*bs+(tx+1)];
        }
```

Figure: Sobel convolution using shared memory and separable filters

```
        __syncthreads();

        // second convolution and write-back
        if (ty>=1 && ty<bs-1 && tx>=1 && tx<bs-1 && y<h && x<w) {
                hKer = tmp2[ty*bs+(tx-1)] - tmp2[ty*bs+(tx+1)];
                vKer = tmp3[(ty-1)*bs+tx] - tmp3[(ty+1)*bs+tx];

                out[y*w+x] = sqrtf(hKer*hKer + vKer*vKer);
                out2[y*w+x] = (byte)((atan2f(vKer,hKer)+9/8*M_PI)*4/M_PI)&0x3;
        }
}
```

Figure: (cont'd.) Sobel convolution using shared memory and separable filters.

```
__global__ void sobel_shm(byte *img, byte *out, byte *out2, int h, int w)
{
        int y, x;
        int vKer, hKer;
        __shared__ int tmp[bs*bs];

        y = (bs-2)*blockIdx.y + threadIdx.y-1;
        x = (bs-2)*blockIdx.x + threadIdx.x-1;

        // load data from image
        if (y>=0 && y<h && x>=0 && x<w) {
                tmp[ty*bs+tx] = img[y*w+x];
        }

        __syncthreads();

        // convolution and write-back
        if (ty>=1 && ty<bs-1 && tx>=1 && tx<bs-1 && y<h && x<w) {
                vKer = tmp[(ty-1)*bs+(tx-1)]*1 + tmp[(ty-1)*bs+tx]*2
                        + tmp[(ty-1)*bs+(tx+1)]*1 + tmp[(ty+1)*bs+(tx-1)]*-1
                        + tmp[(ty+1)*bs+tx]*-2 + tmp[(ty+1)*bs+(tx+1)]*-1;

                hKer = tmp[(ty-1)*bs+(tx-1)]*1 + tmp[(ty-1)*bs+(tx+1)]*-1 +
                       tmp[ty*bs+(tx-1)]*2 + tmp[ty*bs+(tx+1)]*-2 +
                       tmp[(ty+1)*bs+(tx-1)]*1 + tmp[(ty+1)*bs+(tx+1)]*-1;

                out[y*w+x] = sqrtf(hKer*hKer + vKer*vKer);
                out2[y*w+x] = (byte)((atan2f(vKer,hKer)+9/8*M_PI)*4/M_PI)&0x3;
        }
}
```

Figure: Sobel convolution using shared memory and 2D filter

```
__global__ void edge_thin(byte *mag, byte *angle, byte *out, int h, int w)
{
        int y, x, y1, x1, y2, x2;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        // make sure not on the border
        if (y <= 0 || y >= h-1 || x <= 0 || x >= w-1) {
                return;
        }

        // if not greater than angles in both directions, then zero
        switch (angle[y*w + x]) {
        case 0:
                // horizontal
                y1 = y2 = y;
                x1 = x-1;
                x2 = x+1;
                break;
        case 3:
                // 135
                y1 = y-1;
                x1 = x+1;
                y2 = y+1;
                x2 = x-1;
                break;
```

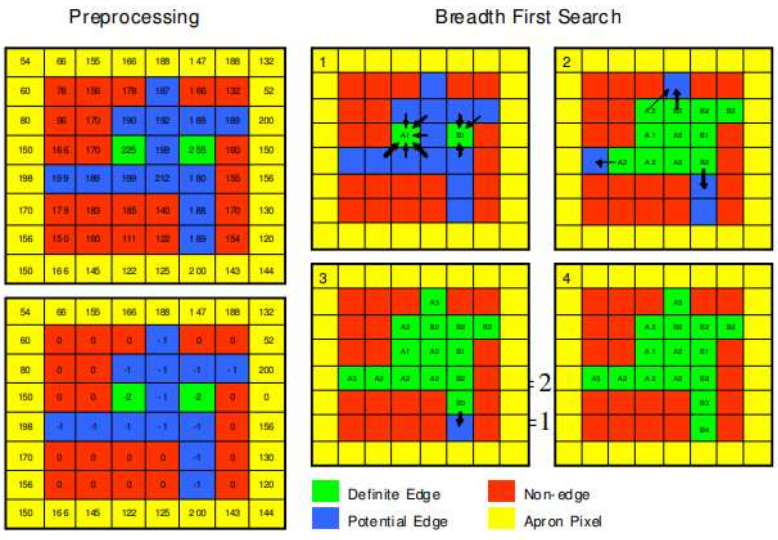
Figure: Edge thinning kernel

```
        case 2:
                // vertical
                x1 = x2 = x;
                y1 = y-1;
                y2 = y+1;
                break;
        case 1:
                // 45
                y1 = y-1;
                x1 = x-1;
                y2 = y+1;
                x2 = x+1;
        }

        if (mag[y1*w + x1] >= mag[y*w + x] || mag[y2*w + x2] >= mag[y*w + x]) {
                out[y*w + x] = 0;
        } else {
                out[y*w + x] = mag[y*w + x];
        }
}
```

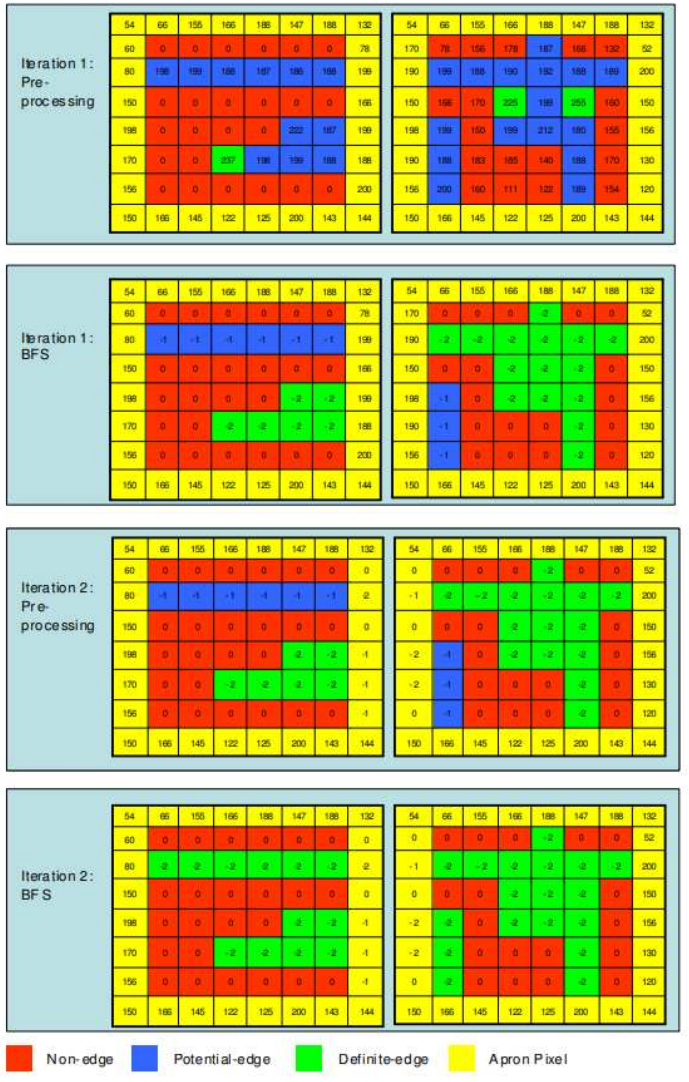
Figure: (cont'd.) Edge thinning kernel

```
#define MSK_LOW          0x0     // below threshold 1
#define MSK_THR          0x60    // at threshold 1
#define MSK_NEW          0x90    // at threshold 2, newly discovered
#define MSK_DEF          0xff    // at threshold 2 and already discovered

// perform double thresholding
__global__ void edge_thin(byte *dImg, byte *out, int h, int w, byte t1, byte t2)
{
        int y, x, ind, grad;

        y = blockDim.y*blockIdx.y + threadIdx.y;
        x = blockDim.x*blockIdx.x + threadIdx.x;

        if (y >= h || x >= w) {
                return;
        }

        ind = y*w + x;
        grad = dImg[ind];
        if (grad < t1) {
                out[ind] = MSK_LOW;
        } else if (grad < t2) {
                out[ind] = MSK_THR;
        } else {
                out[ind] = MSK_NEW;
        }
}
```

Figure: Double-thresholding kernel

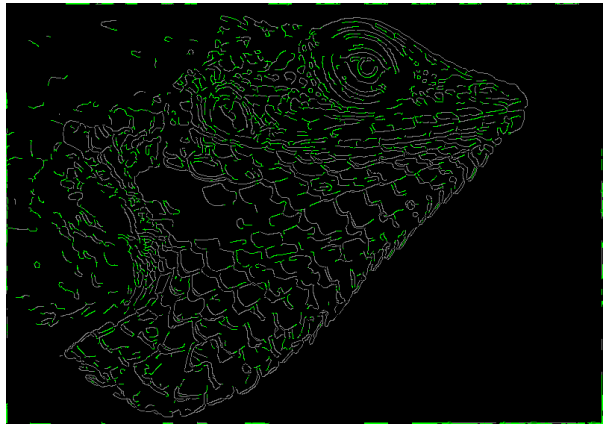


(a) Hysteresis edge tracking within a block (BFS in shared memory)

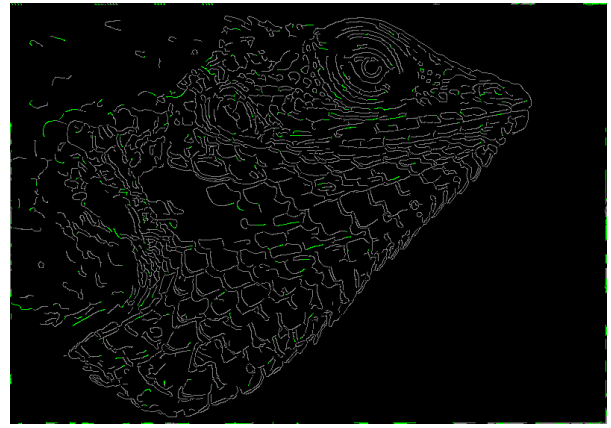


(b) Edge tracking across blocks (multiple passes)

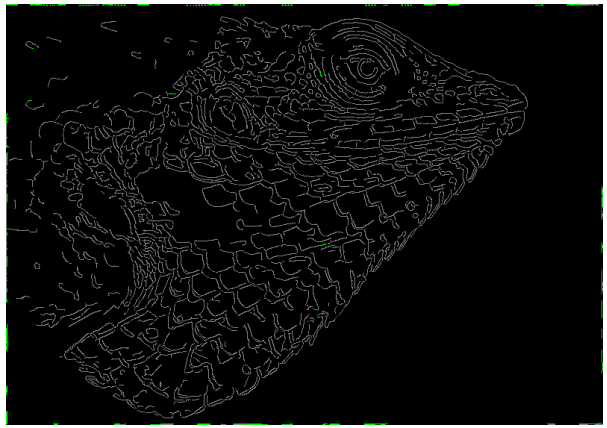
Figure: Hysteresis algorithm. Images source: [2].



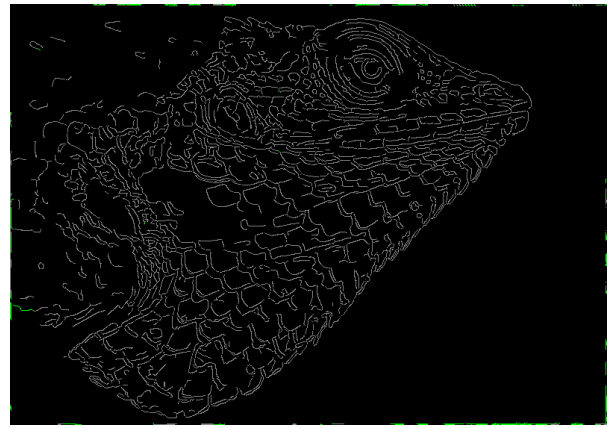
(a) 1 pass



(b) 2 passes



(c) 3 passes



(d) 4 passes

Figure: Effect of varying the number of hysteresis passes on the lizard image. The green pixels indicate new strong edge pixels found in the current hysteresis pass, and gray pixels indicate edges that were already marked strong before the current pass. Very few edges are added after the third iteration. We chose five iterations as the default for our tests.

```
// check and set neighbor
#define CAS(buf, cond, x2, y2, width) \
        if ((cond) && (buf)[(y2)*(width)+(x2)] == MSK_THR) \
                (buf)[(y2)*(width)+(x2)] = MSK_NEW

// perform one iteration of hysteresis
__global__ void hysteresis(byte *dImg, int h, int w, bool final)
{
        int y, x;
        __shared__ byte changes;

        // infer y, x, from block/thread index
        y = blockDim.y * blockIdx.y + threadIdx.y;
        x = blockDim.x * blockIdx.x + threadIdx.x;

        // check if pixel is connected to its neighbors; continue until
        // no changes remaining
        do {
                __syncthreads();
                changes = 0;
                __syncthreads();

                // make sure inside bounds -- need this here b/c we can't have
                // __syncthreads() cause a branch divergence in a warp;
                // see https://stackoverflow.com/a/6667067/2397327

                // if newly-discovered edge, then check its neighbors
                if ((x<w && y<h) && dImg[y*w+x] == MSK_NEW) {
                        // promote to definitely discovered
                        dImg[y*w+x] = MSK_DEF;
```

Figure: Naïve hysteresis using global memory

```
                        changes = 1;

                        // check neighbors
                        CAS(dImg, x>0&&y>0,     x-1, y-1, w);
                        CAS(dImg, y>0,          x,   y-1, w);
                        CAS(dImg, x<w-1&&y>0,   x+1, y-1, w);
                        CAS(dImg, x<w-1,        x+1, y,   w);
                        CAS(dImg, x<w-1&&y<h-1, x+1, y+1, w);
                        CAS(dImg, y<h-1,        x,   y+1, w);
                        CAS(dImg, x>0&&y<h-1,   x-1, y+1, w);
                        CAS(dImg, x>0,          x-1, y,   w);
                }

                __syncthreads();
        } while (changes);

        // set all threshold1 values to 0
        if (final && (x<w && y<h) && dImg[y*w+x] != MSK_DEF) {
                dImg[y*w+x] = 0;
        }
}
```

Figure: (cont'd.) Naïve hysteresis using global memory

```
__global__ void hysteresis_shm(byte *dImg, int h, int w, bool final)
{
        int y, x;
        bool in_bounds;
        __shared__ byte changes, tmp[bs*bs];

        // infer y, x, from block/thread index
        y = (bs-2)*blockIdx.y + ty-1;
        x = (bs-2)*blockIdx.x + tx-1;

        in_bounds = (x<w && y<h) && (tx>=1 && tx<bs-1 && ty>=1 && ty<bs-1);

        if (y>=0 && y<h && x>=0 && x<w) {
                tmp[ty*bs+tx] = dImg[y*w+x];
        }

        __syncthreads();

        // check if pixel is connected to its neighbors; continue until
        // no changes remaining
        do {
                __syncthreads();
                changes = 0;
                __syncthreads();

                // if newly-discovered edge, then check its neighbors
                if (in_bounds && tmp[ty*bs+tx] == MSK_NEW) {
                        // promote to definitely discovered
                        tmp[ty*bs+tx] = MSK_DEF;
                        changes = 1;
```

Figure: Hysteresis using shared memory

```
                        // check neighbors
                        CAS(tmp, 1,             tx-1, ty-1, bs);
                        CAS(tmp, 1,             tx,   ty-1, bs);
                        CAS(tmp, x<w-1,         tx+1, ty-1, bs);
                        CAS(tmp, x<w-1,         tx+1, ty,   bs);
                        CAS(tmp, x<w-1&&y<h-1, tx+1, ty+1, bs);
                        CAS(tmp, y<h-1,         tx,   ty+1, bs);
                        CAS(tmp, y<h-1,         tx-1, ty+1, bs);
                        CAS(tmp, 1,             tx-1, ty,   bs);
                }

                __syncthreads();
        } while (changes);

        if (y>=0 && y<h && x>=0 && x<w) {
                if (final) {
                        if (in_bounds) {
                                dImg[y*w+x] = MSK_DEF*(tmp[ty*bs+tx]==MSK_DEF);
                        }
                } else {
                        dImg[y*w+x] = max(dImg[y*w+x], tmp[ty*bs+tx]);
                }
        }
}
```

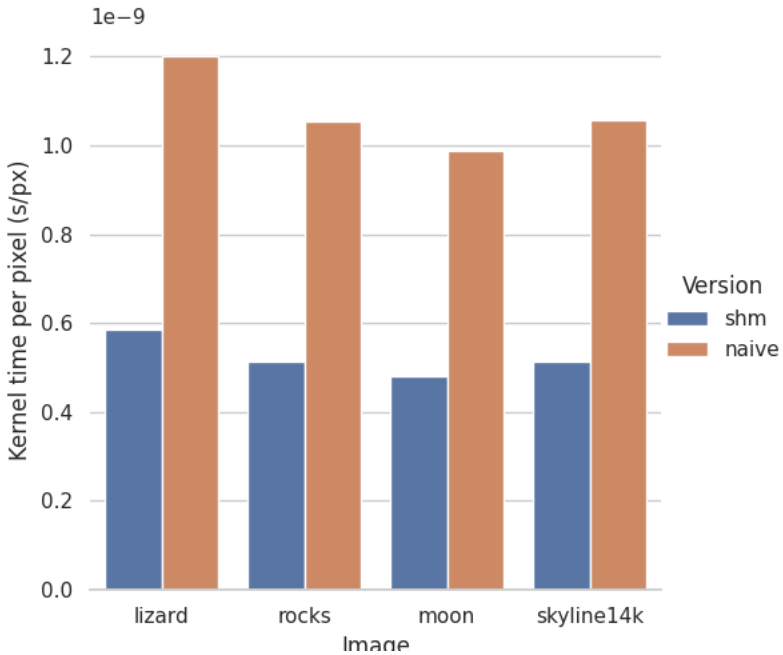
Figure: (cont'd) Hysteresis using shared memory

Test setup and objectives:

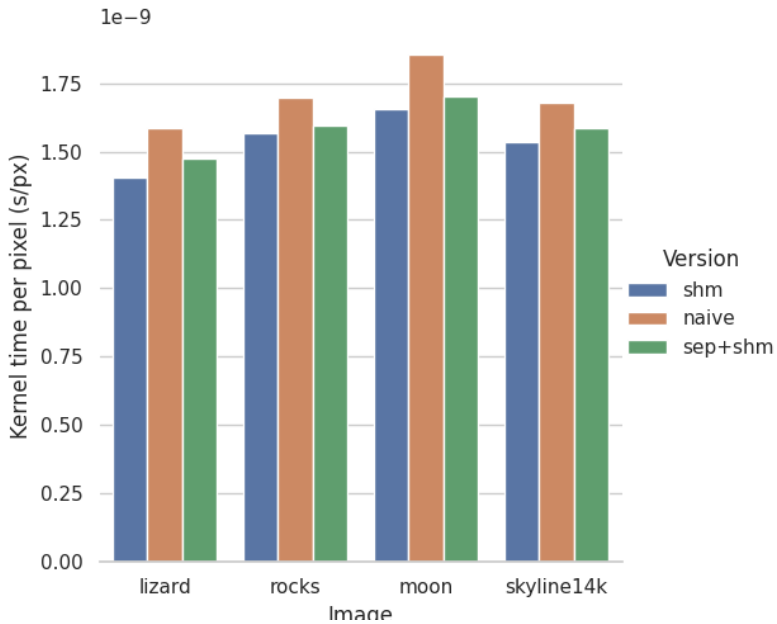
- NVIDIA GT740 (1GB) vs. Intel i5-7267U
- Compare all kernels with CPU equivalents
- Compare optimized/unoptimized kernels
- Compare accuracy

Kernel	Time Reduction (%)	Speedup (%)
blur	99.98 (opt); 99.3 (unopt)	500000; 14200
sobel	96	2400
edgethin	93	1300
threshold	86	600
hysteresis	29	40

Table: Speedups for CUDA kernels vs. CPU equivalent. Average image accuracy: $\approx 99\%$.

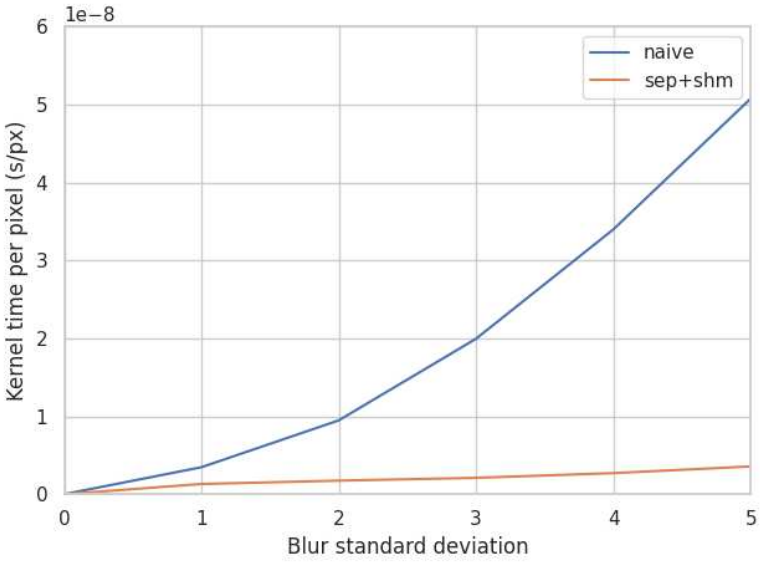


(a) Comparison of normalized kernel times of the two hysteresis implementations.

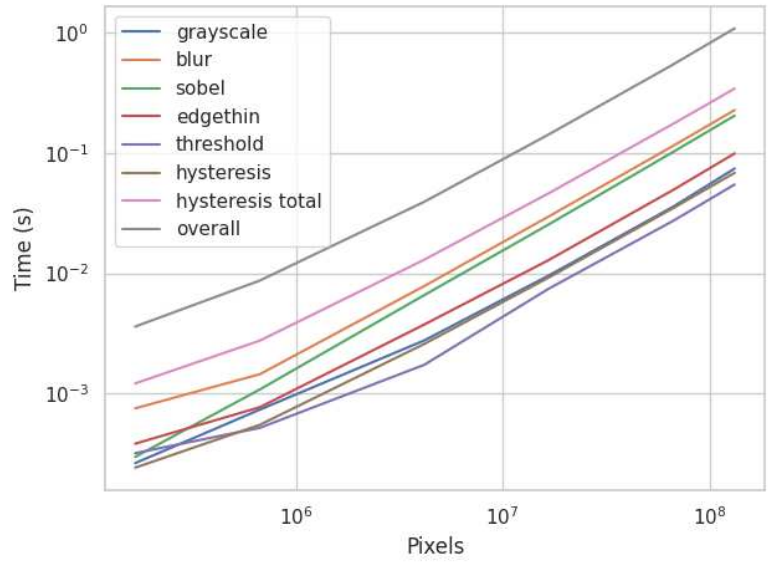


(b) Comparison of normalized kernel times of the three Sobel filter implementations.

Figure: Result of optimizing kernels. Plots are normalized by the number of pixels in the image.



(a) Comparison of the normalized kernel times of the two (Gaussian) convolution implementations for varying σ.



(b) Timings of the different kernels on images of different sizes.

Figure: Showing scalability of the two blur kernels for different blur sizes, and for all kernels over different image sizes.

References

Victor Podlozhnyuk.
Image convolution with cuda.
NVIDIA Corporation white paper, June, 2097(3), 2007.

Yuancheng Luo and Ramani Duraiswami.
Canny edge detection on nvidia cuda.
In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.

ECE453 – Deblurring

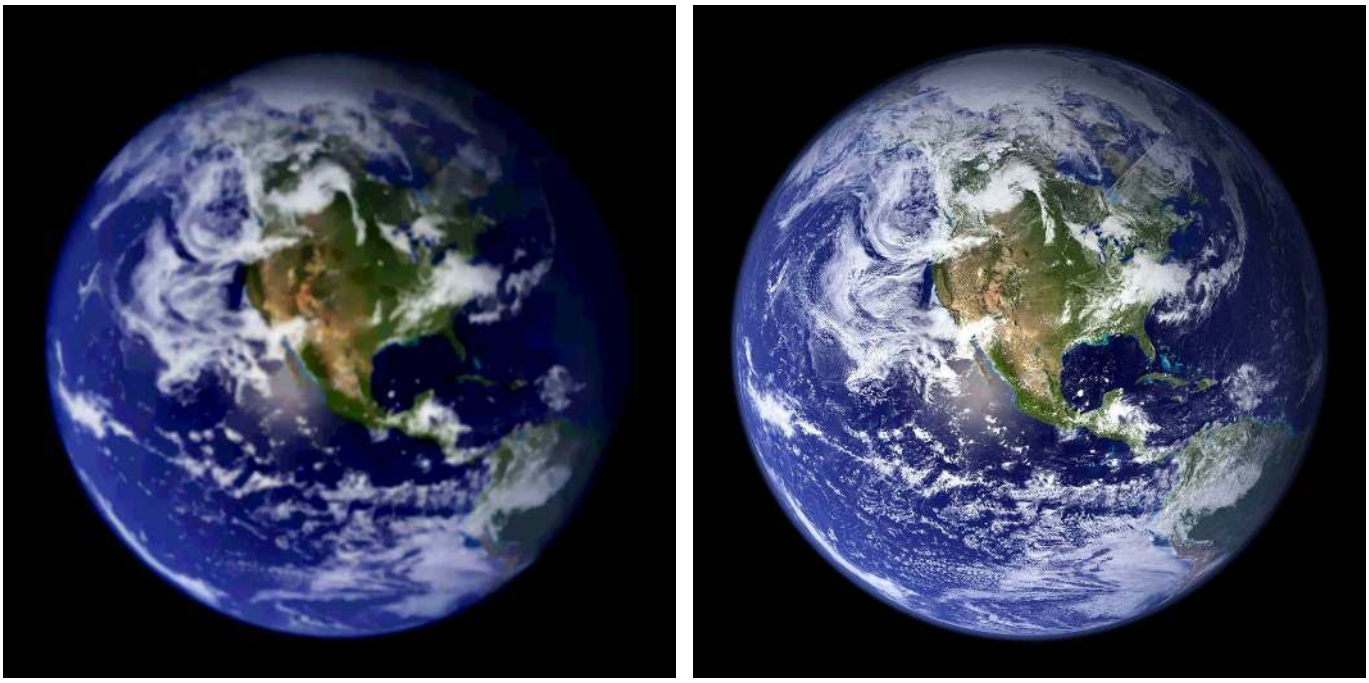
Jonathan Lam

March 21, 2021

1 Introduction

The goal of this project is to become familiar with the CUDA toolchain and use a highly-parallelized CUDA program to achieve a speedup over a traditional CPU program.

The task is to unblur the image. A clear reference image is also provided.



(a) Blurry test image

(b) Clear reference image

Figure 1: The provided images. Both are 960x960 PNG images.

2 Theory

This is not a comprehensive survey of the theory, since this is only an introductory project and the main goal was to learn CUDA, not the theory in full. The full theory is deferred to the research papers mentioned in the following sections.

2.1 Deblurring algorithm

The general idea is that a typical blurred image can be modeled as the convolution of the original image with some distorting filter (this filter is called the point spread function, or PSF). This filter can be used to model blurring caused by factors such as defocus aberration (when an image is slightly out of focus) or motion blur.

Thus the problem is broken into two parts: how to find the PSF, and, once it is known, how to deconvolve the blurry image with the PSF.

The method chosen is the Richardson-Lucy (RL) deconvolution algorithm [1, 2], which, given a known PSF, takes an iterative approach that is guaranteed to converge to the maximum-likelihood deconvoluted image. Fish et. al. developed a method to swap the roles of the original image and the PSF [3] to find the maximum-likelihood estimate for the PSF given the convolved image and the "known" (i.e., estimated) deconvolved image. This can be used iteratively as a blind deconvolution method to get maximum-likelihood estimates of both the PSF and the original deconvolved image by alternatively fixing one and estimating the other.

A paraphrasing of Fish et al's description of the reasoning behind the algorithm in layman's terms goes as follows: by Bayes' theorem, we have:

$$P(x_i \mid y) = \frac{P(y \mid x_i)P(x_i)}{\sum_j P(y \mid x_j)P(x_j)}$$

We can interpret this (very roughly) as: $P(x)$ being the true distribution (i.e., the pristine image); $P(y)$ being the convolved distribution (the blurry image); $P(y \mid x)$ is the PSF centered at x. Note that $P(x_i)$ appear on both sides of the equation – we can use this to improve our estimate using the previous estimate. By multiplying by $P(y)$ and rearranging terms we obtain:

$$P(x_i) = \sum_k P(x_i \mid y_k)P(y_k) = \sum_k \left[\frac{P(y_k)}{\sum_j P(x_j)P(y_k \mid x_j)} P(y_k \mid x_i) \right] P(x_i)$$

The additional sum is added because the pixel at x is based on a range of pixels in the deconvolved image. The summations translate to convolutions, and this can be written as:

$$f_{i+1}(x) = \left\{ \left[\frac{c(x)}{f_i(x) \otimes g(x)} \right] \otimes g(-x) \right\} f_i(x)$$

following Fish et al's notation.

The "blind" part of the algorithm, i.e., estimating the PSF works the same way, but swaps the roles of g (the PSF) and f (the image). The formula as stated in Fish et al. is

$$g_{i+1}(x) = \left\{ \left[\frac{c(x)}{g_i(x) \otimes f(x)} \right] \otimes f(-x) \right\} g_i(x)$$

Now that we have a method of solving for the (next iteration of) the deblurred image and the (next iteration of) the PSF, we alternate between solving for one and then the other.

I was originally going to try the method used in Fish et al. to do the full blind deconvolution (i.e., also estimating the PSF), but this proved too time-consuming a task, so I stuck with the ordinary RL method using a fixed (chosen) PSF. For this implementation, circular (2D) Gaussian filters with various standard deviations were tested, and for the given test image a standard deviation of roughly $\sigma = 3$ appeared to produce the best results. The filter dimension used for this value of σ is roughly $\lceil 6\sigma \rceil$ so as to not lose too much power (i.e., so that the filter roughly has unity norm).

2.2 Evaluation

To evaluate correctness of the algorithm, a program was written to calculate the elementwise squared error between two images (where the "element" is each data channel of each pixel). This is used both to calculate the error between the blurry and deblurred images and the original (to show that deblurring does not make the image substantially less faithful to the original) and between the CPU and CUDA versions (to show that the two implementations are equivalent in output).

To evaluate the effectiveness of the deblurring, a method similar to that in an answer by @Niki on Stack Overflow that suggests convolution of the image by a Laplacian of Gaussian (LoG) filter. This filter is essentially an edge detection filter and will have values of larger magnitude near areas with a steeper gradient. While @Niki's use suggests the idea of a "general blurry" metric that works across dissimilar images, we are only comparing similar images; as a result, I chose instead to use the average value of the image convolved with the LoG filter (as opposed to the 99.9th percentile value) to get a better estimate of the relative blurriness across the image.

3 Implementation

The source code for this project is located at `https://github.com/jlam55555/ece465-adv-comp-arch`.

3.1 CUDA version

The general algorithm flow involves: loading in the image to a (contiguous) buffer; stripping alpha values (for PNG images); converting integers to floats; allocating space of the same size on the GPU for the image buffer, as well as temporary buffers for intermediate values; iterating over the RL algorithm; copying the image back from device memory; and writing the image to a file.

The RL algorithm is fairly straightforward (requiring two convolution operations and two elementwise multiplication/division operations). The convolution takes two arguments (an input image and a filter) and outputs to a third buffer; the output has the same size as the input image (because all operands in the algorithm must be the same size for the pointwise multiplication to work), and the filter must be centered at index zero so that the convolution does not shift the original image. This requires bounds checking and a transformation of the filter's coordinates to zero-centered coordinates.

There are four (CUDA) kernels used in total: two to convert images to and from bytes and floats (and also strip/restore the alpha value); the 2D convolution; and the elementwise multiplication/division.

3.2 CPU version

The CPU-only version is very similar to the CUDA version, and the data types are identical (all calculations are done in 32-bit floating-point and cast back to 8-bit pixel values). Thus we expect a very small error between the outputs of the two algorithms, and this is indeed the case.

The code is somewhat shorter because there is no need to create separate host- and device-versions of the input buffer.

3.3 Benchmarking and image I/O

A very simple benchmarking implementation was created using the standard library's `time.h` utilities. It allows the user to create several timer objects with a stopwatch "lap" function, and stores the average time taken for different operation types in an array.

There was some trouble getting this to work on the CUDA implementations, since the CUDA kernels run asynchronously to the C code – this required the use of the `cudaDeviceSynchronize()` method. This likely adds a small but unnoticeable decrease in the CUDA implementation's performance. While this worked on my local machine, this method did not successfully keep track of time on the Jetson Nano for unknown purposes – this requires future investigation.

For the time being, a less accurate estimate of the time taken is found using the Unix `time` shell command.

Image I/O was handled through the use of the libpng library using sample I/O code by Guillaume Cottenceau. This was chosen because libpng is fairly standard and easy to install (as opposed to more capable libraries like ImageMagick's C/C++ API). However, this means that only PNG images are supported in this version.

3.4 Evaluation of error/correctness

This was implemented quite simply by looping through all pixels of two images and calculating the mean squared error between each corresponding data value (excluding the alpha channel).

3.5 Evaluation of blurriness

The bluriness metric was implemented by convolving (reusing the 2D convolution function) the input image with a 3×3 LoG kernel, and then taking the mean squared value of the convolved image.

4 Results

See the GitHub repository README for detailed build and run instructions.

The implementations were tested on a local machine and the school's Jetson Nano. The local machine has an Intel i7-2600 CPU (x86_64 architecture, up to 3.8GHz single-core clock speed) with a Nvidia GT740 (Kepler architecture, 384 CUDA cores) GPU. The Nano has a Cortex-A57 CPU (1.4GHz, aarch64, up to 1.4GHz) with a Maxwell architecture, 128 CUDA cores GPU.

The performance results of running both the CPU and GPU (CUDA) versions with 25 iterations and the best value of σ are shown in Table 1. Performance depends on the exact model of GPU and CPU, but there is a huge gap between the performance of the two CPUs and those of the two GPUs. For the two CPU-GPU pairs on the test systems, there are 37× and 110× speedups. (The speedup on the Nano is expectedly larger, as this is a machine dedicated to GPU-heavy tasks). Most of the time and computation spent in each iteration occurs in the convolutions, and this is also where the large speedup occurs.

The MSE between the CPU and CUDA version for $\sigma = 3$ is 0.000014, which is small enough to be unrecognizable. Thus we conclude the two are equivalent.

The MSE between each image and the clear reference image, as well as the sharpness metric, is shown in Table 2. Increasing σ monotonically decreases the fidelity of the deblurred image, but this is fine as we don't expect a very good result if the image was initially blurred with a large radius. All that matters for the MSE is that it is sufficiently close to that of the input blurry image.

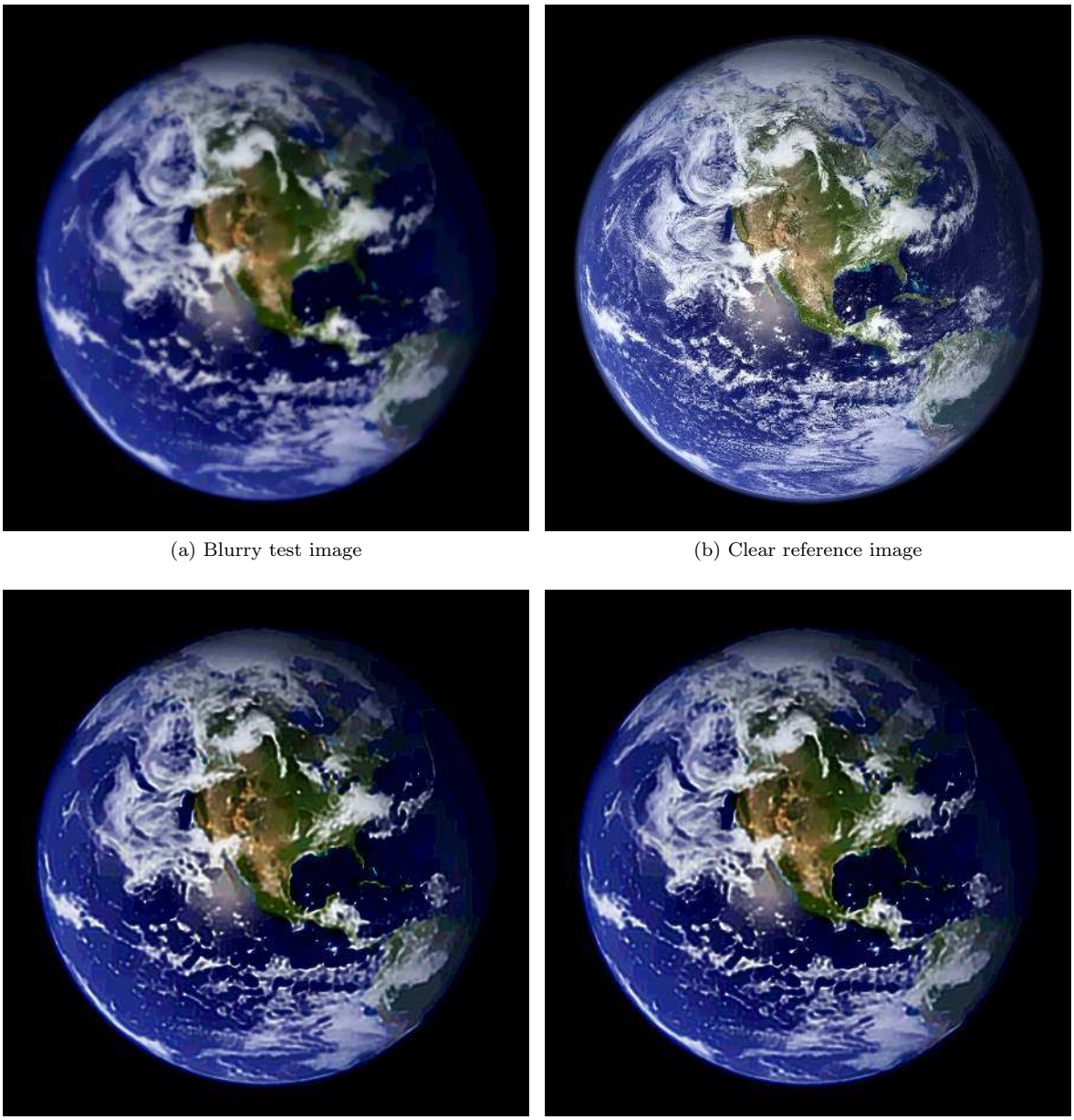
The sharpness is not monotonic with respect to σ. The best σ should be that which best matches the gaussian estimate of the original PSF.

	i7-2600 (CPU)	**Cortex-A57 (CPU)**	**GT740 (CUDA)**
Overall	219	880	5.93
RL Iteration	8.76	35.2	0.237
2D Conv kernel	4.37	17.6	0.114
Mult/Div kernel	0.0107	0.0350	0.00438

Table 1: Average time (s) of operations on the CPU and GPU implementations. Tests were performed with $\sigma = 3$ (19×19 filter). Timing did not work correctly on the Nano CUDA implementation; a rough measurement taken is 8s overall.

	Original	**Blurry**	**CPU**	**CUDA**			
			$\sigma = 3$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$\sigma = 4$
MSE	0	1247	1380	1264	1274	1380	1538
Sharpness	1695	8	53	23	37	53	51

Table 2: MSE and sharpness values, as defined in 2.2, of the input (clear and blurry) image and the input images.



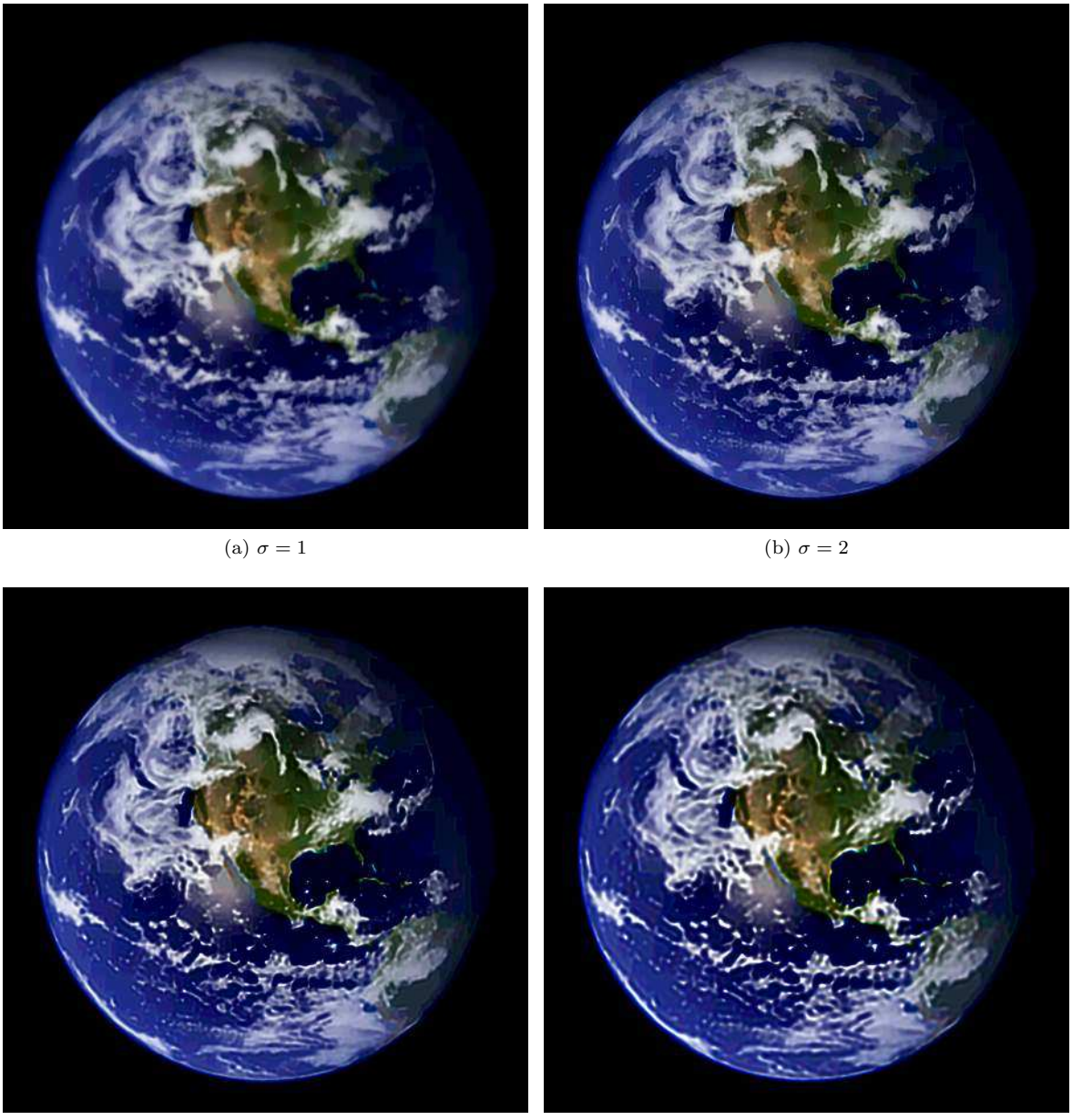
(a) Blurry test image

(b) Clear reference image

(c) CUDA deblurring

(d) CPU deblurring

Figure 2: The provided images vs. the deblurred images with 25 iterations and $\sigma = 3$. The CUDA and CPU versions are indistinguishable. (The difference is best seen on a large screen with the images on GitHub.)



(a) $\sigma = 1$

(b) $\sigma = 2$

(c) $\sigma = 3$

(d) $\sigma = 4$

Figure 3: Deblurred images with 25 iterations of RL using the CUDA implementation, but with different values for σ. The image gets slightly clearer going from $\sigma = 1$ to $\sigma = 2$ and from $\sigma = 2$ to $\sigma = 3$, but excessive ringing begins to appear at $\sigma = 4$ (which likely contributes to the decrease in the sharpness.

5 Conclusions

The programs were fairly successful at demonstrating the speedup that GPUs can offer. This project was also successful in getting the student more familiar with writing kernel functions in CUDA rather than using a loop-based or CPU multi-threaded approach. It was also informative learning about blind deconvolution, the RL-method, and LoG filters.

References

[1] L. B. Lucy. An iterative technique for the rectification of observed distributions. *Astronomical Journal*, 79:745, June 1974.

[2] William Hadley Richardson. Bayesian-based iterative method of image restoration∗. *J. Opt. Soc. Am.*, 62(1):55–59, Jan 1972.

[3] D. A. Fish, A. M. Brinicombe, E. R. Pike, and J. G. Walker. Blind deconvolution by means of the Richardson-Lucy algorithm. *Journal of the Optical Society of America A*, 12(1):58–65, January 1995.

A Serverless Convex Hull Web Application

ECE465 Final Project

Jonathan Lam

The Cooper Union for the Advancement of Science and Art

May 13, 2021

Why convex hull web app?

Why convex hull?

- Convex hull is a useful algorithm, e.g., for computer vision
- Originally planned to do HPC, convex hull is a divide-and-conquer algorithm

Why serverless?

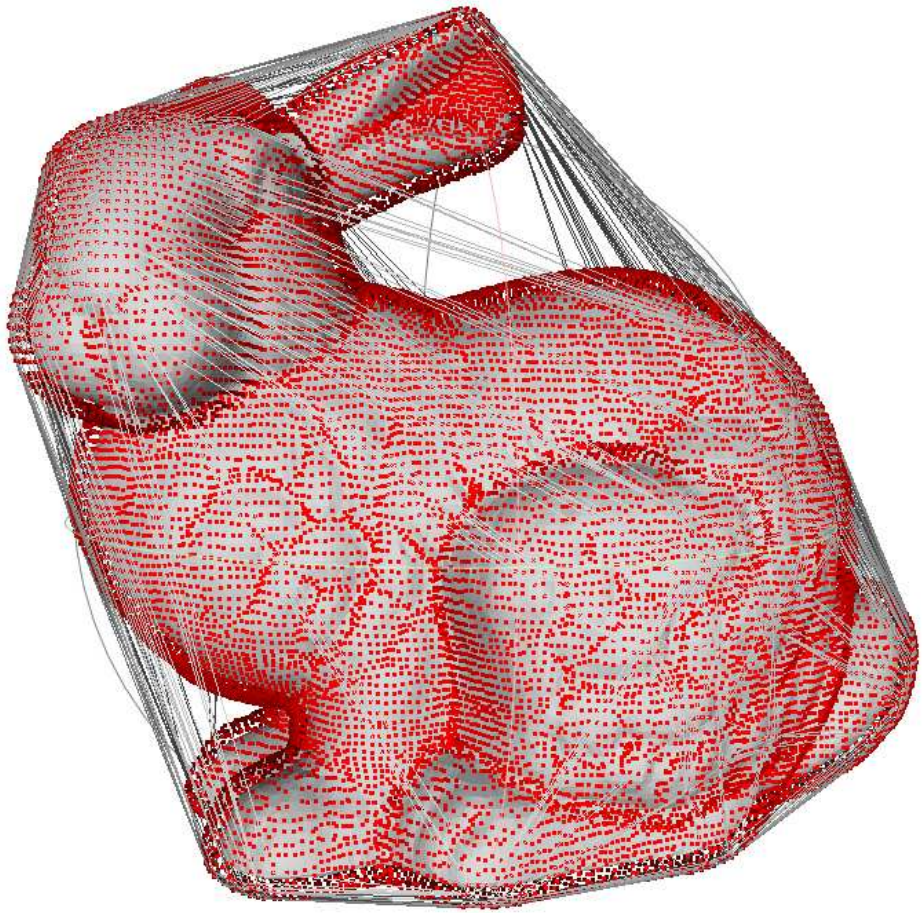
- Competitive pricing to traditional servers
- Better decoupling/maintainability of services

Project trajectory

custom HPC distributed 3-D qhull algorithm

web app using existing 3-D qhull library

(3-D) Convex hull



(3-D) Convex hull

Implementation difficulties

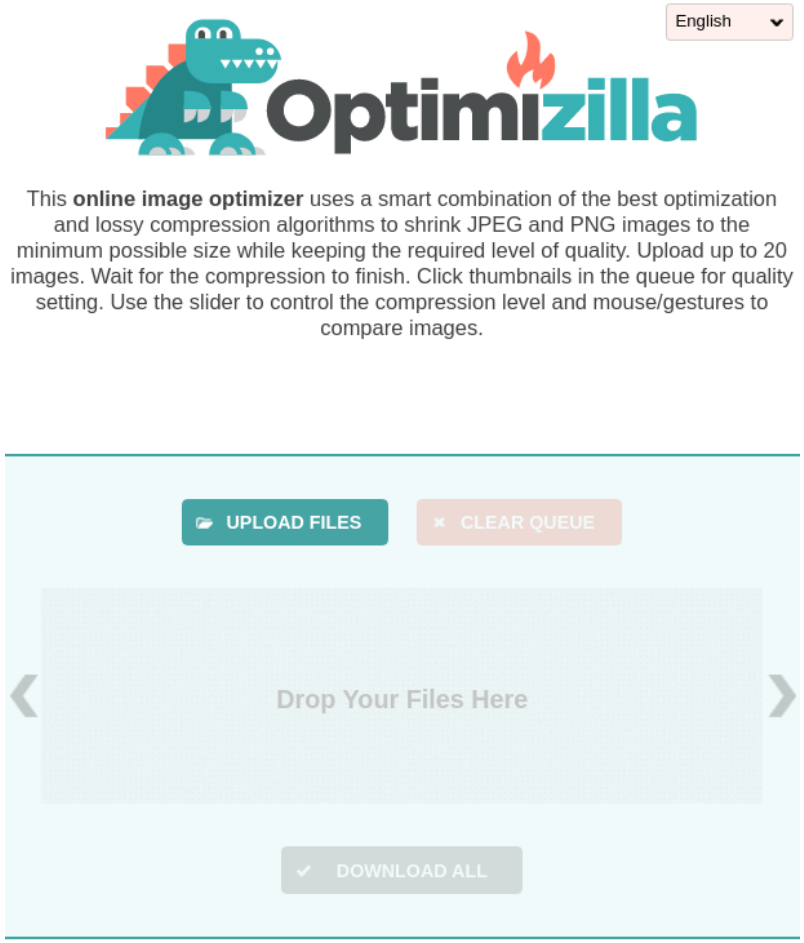
- Handle degenerate cases
- FP numerical precision issues
- Need to merge coplanar faces when they emerge to avoid problematic geometries: "thick plane" coplanar test
- Non-triangular faces due to face merging
- Newell's method for arbitrary (planar) polygonal normals – more robust to nonplanarity
- Half-edge data structure
- DFS over visible faces to find horizon
- Have to resolve possible topological errors on face merging

(3-D) Convex hull resources

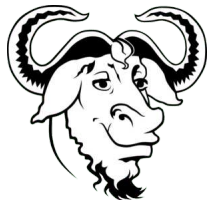
- "Implementing Quickhull – Dirk Gregorius" – an amazing visual tutorial on 3-D QuickHull
- qhull – a well-known robust QuickHull (and related algorithms) implementation
- A robust and academic QuickHull3D implementation in Java

Vision of finished product

Something like Optimizilla – one-off image processing



Tech stack

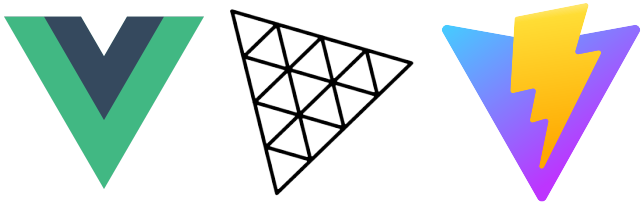


```
aws-cli/2.1.32
```

Build

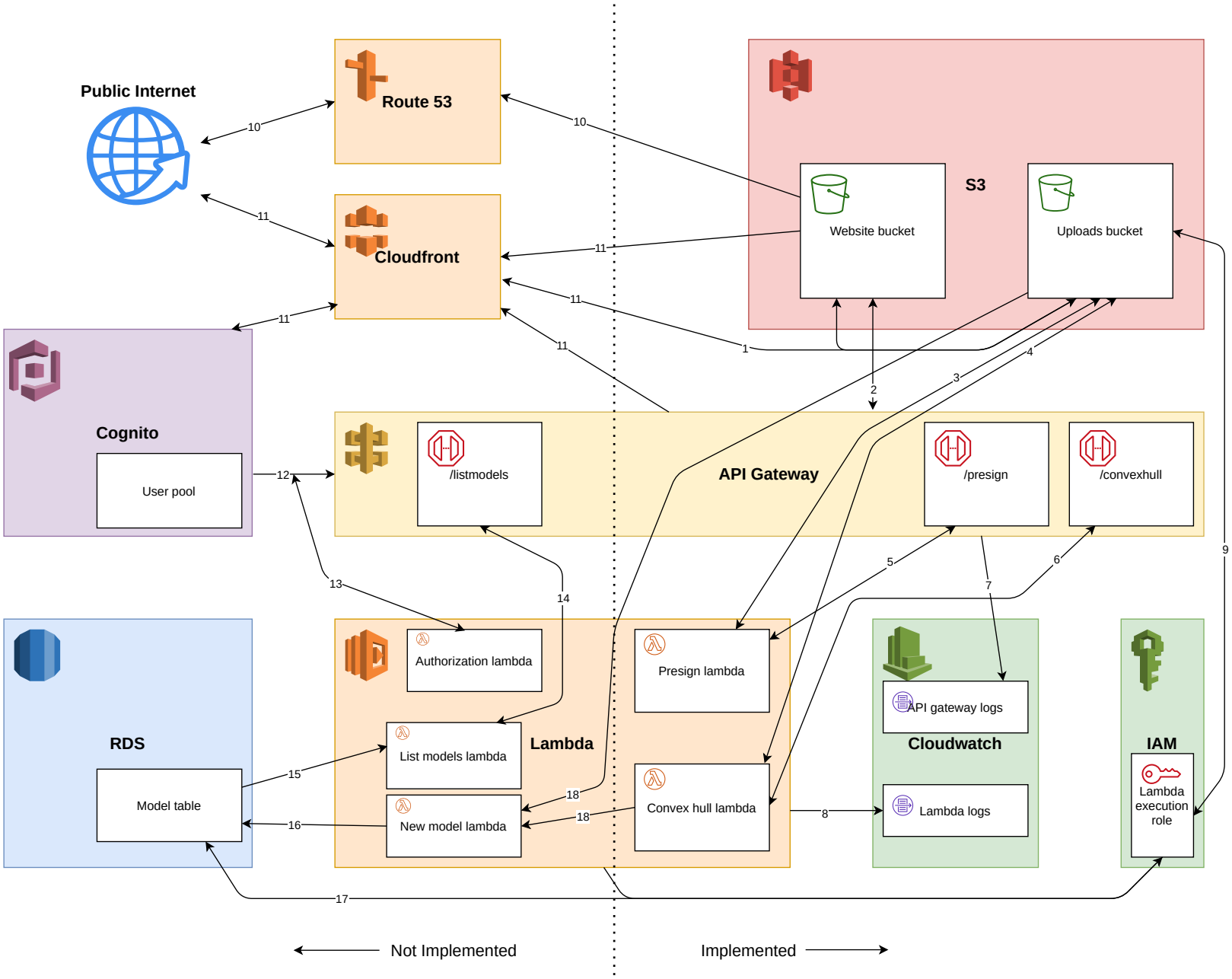
```
github.com/aws/aws-lambda-go/lambda
github.com/aws/aws-sdk-go/service/s3
github.com/markus-wa/quickhull-go
```

Lambdas



Frontend

Architecture diagram



Architecture diagram notes

1. Access to uploads bucket by presigned URLs only.
2. Appopriate CORS headers for website use.
3. Presign lambda has read access to uploads bucket.
4. Convex hull lambda has GET/PUT access to uploads bucket.
5. Presign endpoint calls presign lambda.
6. Convex hull endpoint calls convex hull lambda.
7. API Gateway stage has logging set up.
8. Each lambda has logging set up (to different loggroups).
9. Lambdas need access to uploads bucket.
10. Route 53 acts as a DNS server.
11. Cloudfront acts as a CDN for the website, uploads bucket, API Gateway, and authorization page. It also supports HTTPS (necessary for Cognito).
12. Cognito authorizes API requests using the users access token.
13. Authorization lambda verifies the user's access token.
14. Listmodels endpoint calls listmodels lambda.
15. Listmodels lambda retrieves models entries from database.
16. New model lambda creates new model entry in database.
17. Listmodels/Newmodel lambdas require permissions for RDS.
18. On new upload (PUT request) or creation of a hull (a new model), newmodel lambda will trigger

Components

Front-end:

- Vue 3 framework and Vite build tool
- THREE.js for loading/display OBJ files
- Static web hosting using S3

API:

- CORS policy for front-end, lambda policy
- Endpoints: /presign, /convexhull

Back-end (lambdas):

- Custom OBJ file loader/dumper (package objio)
- Uses package markus-wa/quickhull-go for 3-D quickhull
- Uses AWS SDK (aws-sdk-go-v2) for lambda and s3 utilities
- Implements Lambda proxy integration

Components, cont'd.

Build system:

- Makefile with a lot of macros
- Uses aws-cli v2

Buckets:

- Allow public access via pre-signed GET and PUT requests
- Static (public) web hosting for front-end with index and error page

Things I wanted to try (but ran out of time to implement)

- Cognito user pools
  - OAuth2 Code Authorization Flow (w/ external identity providers)
  - Can allow authorization of API endpoints (like API key) using Access Token
  - Not too familiar with proper handling and verification of JWT tokens and Identity/Access/Refresh tokens
  - Requires HTTPS (Route 53)
- RDS
  - Store user and model information
- Route 53 and CloudFront
  - For DNS, HTTPS, and CDN services
  - Not on AWS Educate

Build process

Top-level targets in the Makefile:

```
.PHONY:
all: build-website\
        host-bucket-create\
        host-bucket-sync\
        upload-bucket-create\
        upload-bucket-policy-create\
        lambda-iam-create\
        loggroup-create\
        lambda-create\
        api-create

.PHONY:
clean: target-clean\
        api-delete\
        lambda-delete\
        loggroup-delete\
        lambda-iam-delete\
        upload-bucket-policy-delete\
        upload-bucket-delete\
        host-bucket-delete
```

Demo

Demo link

Home

Upload a file to begin

Choose File | No file chosen | Upload file

Home

Viewing model: 3cq5bgoB

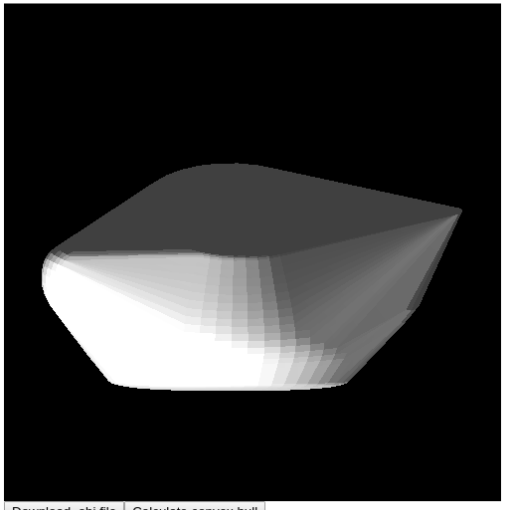
100.00% loaded

Download .obj file | Calculate convex hull

Home

Viewing model: zezqPQl0

100.00% loaded



Download .obj file | Calculate convex hull

Lessons learned

To be more efficient:

- Use Amplify (Cognito/Route 53/Cloudfront integration)
- Use CDK (no shell/Makefile problems, full general-purpose language logic)

(but still a good learning experience w/o them)

Questions?

(Source code available on GitHub at @jlam55555/cloud-convex-hull.)

ECE465 Final Project Proposal

Jonathan Lam & Henry Son

March 1, 2021

Motivation for changing our algorithm

Over the course of the first project, we realized that our previous "distributed" algorithm (graph coloring) induced too high of a communication cost and was not very amenable to speedup over networked nodes. As a result, we decided to change our focus to a problem that is more cleanly divisible into parts (i.e., one with low communication costs) for sub-processing between nodes.

What is convex hull?

Given a set of points P, the convex hull of P is the smallest convex shape that contains all the points within P. It has numerous uses in geometric modeling, statistics, and etc. Practical use cases include collision avoidance, smallest bounding box, and shape analysis.

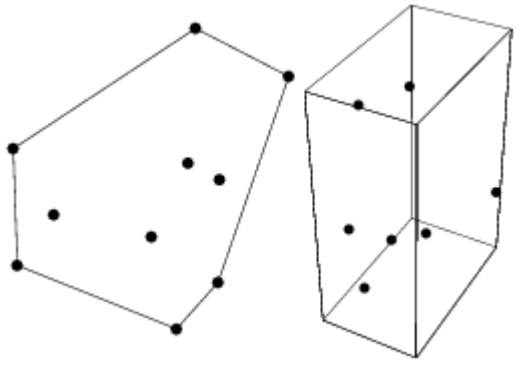


Figure 1: 2D, 3D examples of Convex Hulls. As you can see from the diagram, the pointset is contained within the convex hull, denoted as the area/volume enclosed by the solid lines.

Quickhull: an algorithm for finding the convex hull

Quickhull is a divide-and-conquer algorithm and can be used for arbitrary N-dimensional space. A good explanation, as well as pseudocode for the 2-D case, can be found on its Wikipedia page. The input pointset will first be divided into non-overlapping regions to be distributed among compute nodes, which will each perform Quickhull independently. These disjoint convex hulls will then be joined on the coordinator node.

What it'll look like

The final project might look like an API that has endpoints to receive 2-D or 3-D models (or arbitrary N-D models) and calculate their convex hulls. For example, a user could input a file generated by a 3-D CAD program and receive a "gift-wrapped" version of it (its convex hull). Time permitting, there may also be additional endpoints related to convex hulls, such as calculating diameter of a convex hull, finding the union or intersection of convex hulls, distance between convex hulls, etc.

Infrastructure, environment, and tooling

The infrastructure would be an AWS Lambda[1] compute to reduce costs and easily scale. The algorithm will again be written in Go and use the default Go build tools.

[1] Neither of us have used Lambda before nor know exactly how it works, so it may end up being on regular EC2 instances.

ECE465 Checkpoint 2b
Distributed Graph Coloring

Jonathan Lam & Henry Son

February 21, 2021

Since the diagrams were written with TikZ, it was easier (and prettier) to typeset everything in LaTeX than convert and embed all the visuals into the GitHub Markdown files. I'll probably end up all of the reports (i.e., everything but basic build instructions) over to LaTeX because it looks nicer and has a more academic feel.

1 Updates from Checkpoint 2a

The handshake was left mostly unchanged, as it was already working fine for the first part. The algorithm was barely functional for the first part of the assignment, so the main goal was to fix up these errors. This was two-fold: fixing a issue that involved short writes, and synchronizing the start of an iteration in the algorithm in a way that avoids a disastrous race condition.

Performance-wise, we did not have enough time to achieve a speedup over single-node application. We spent much time on debugging (enough to start jeopardizing other schoolwork) and were only barely able to finish making the application run reliably.

There was also some general cleanup of code and adding additional features, such as a quiet flag to suppress logging, and disabling printing of nodes (otherwise the 10000 node graph generates a 2.5GB logfile). More details on the fixes and performance issues follow the figures below.

2 Figures

See the following pages for details on the entire distributed algorithm. Note that the terms "client" and "worker" are used interchangeably and refer to all of the non-server nodes.

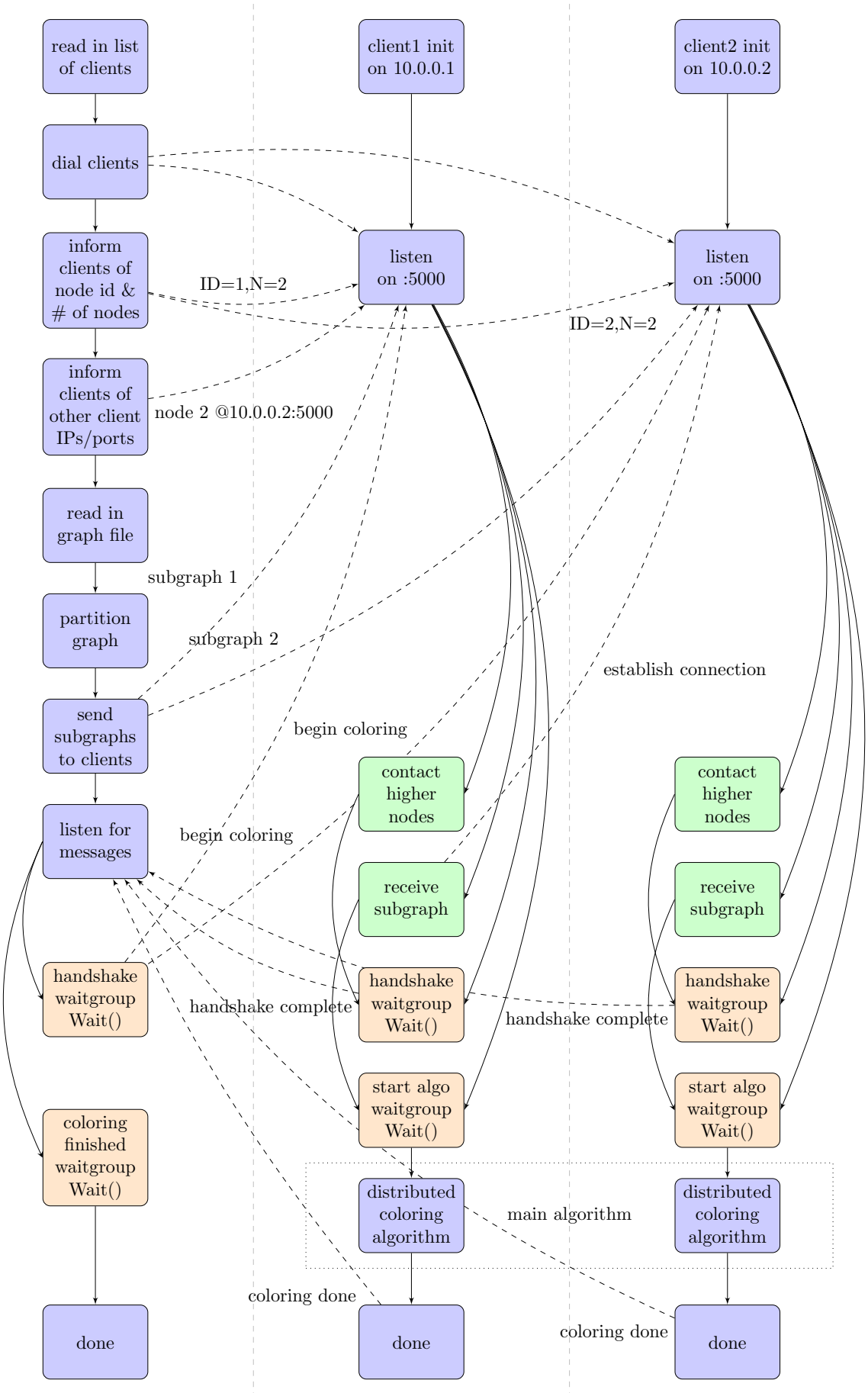


Figure 1: Communication protocol for the coloring algorithm for a two-worker network. Solid and dashed lines represent intra- and inter-node (socket) communication, respectiely. Blue blocks represents sequential code, green represents handlers to socket messages, and orange represents waitgroup (semaphore) action.

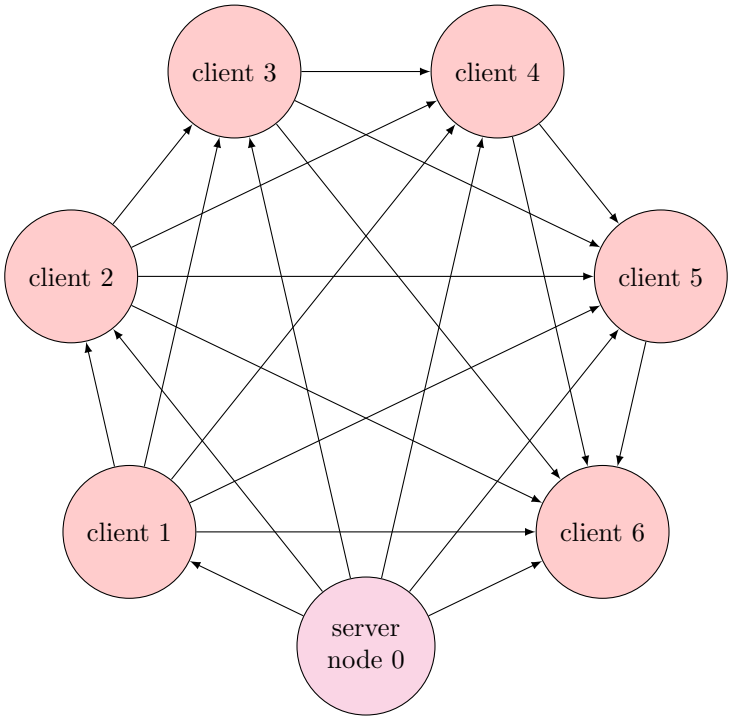


Figure 2: Illustration of the node dialing order for a six-worker network. The server ("node 0") dials each worker node, informing them of their node ID (1-6), total number of nodes (7), and the addresses (IP:port) of higher-indexed nodes. Each node then dials each of its higher-ordered nodes until a complete graph of peer-to-peer (TCP socket) connections is formed. (This process is relatively brief, so the imbalance is inconsequential.) Only then is the handshake complete.

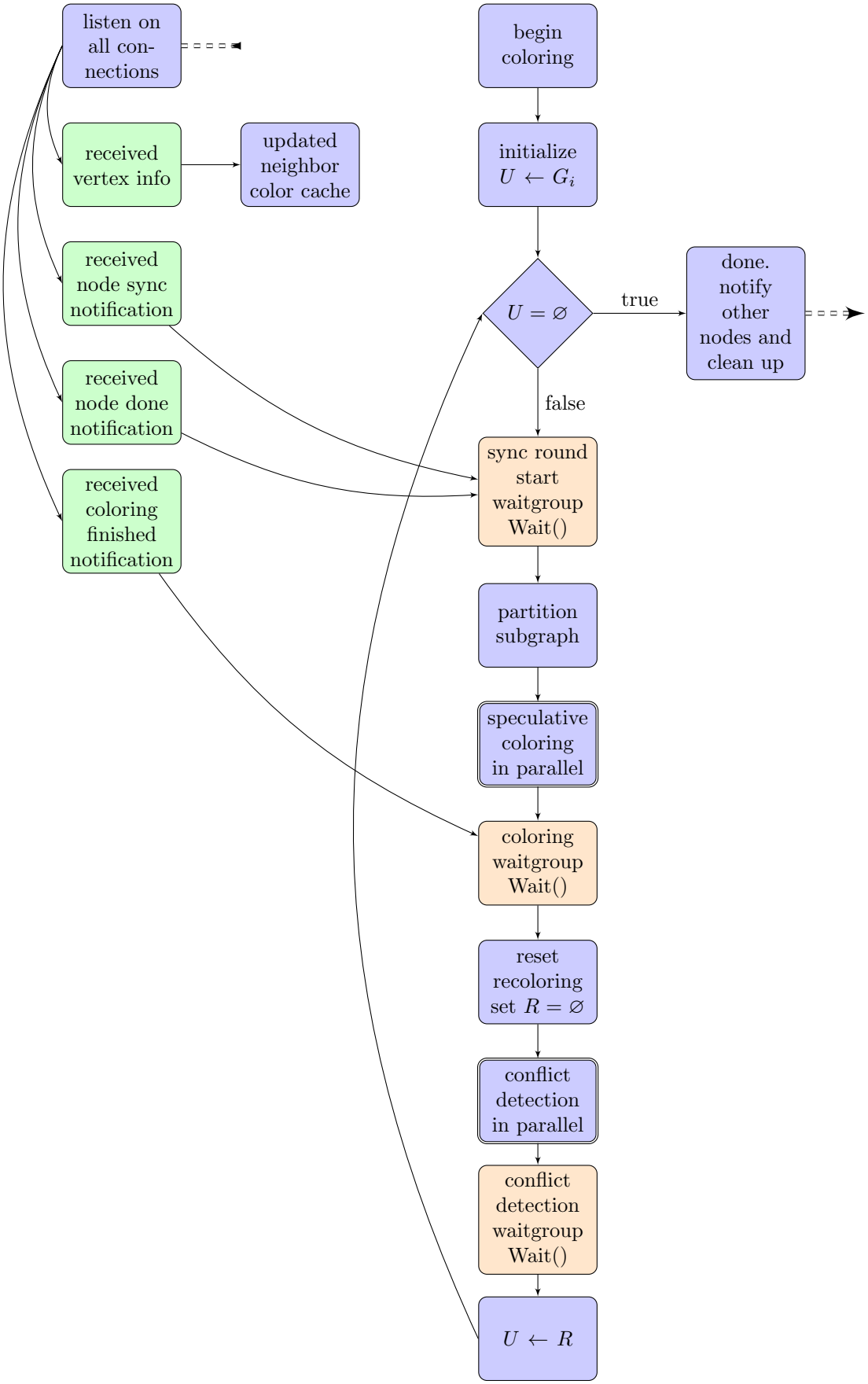
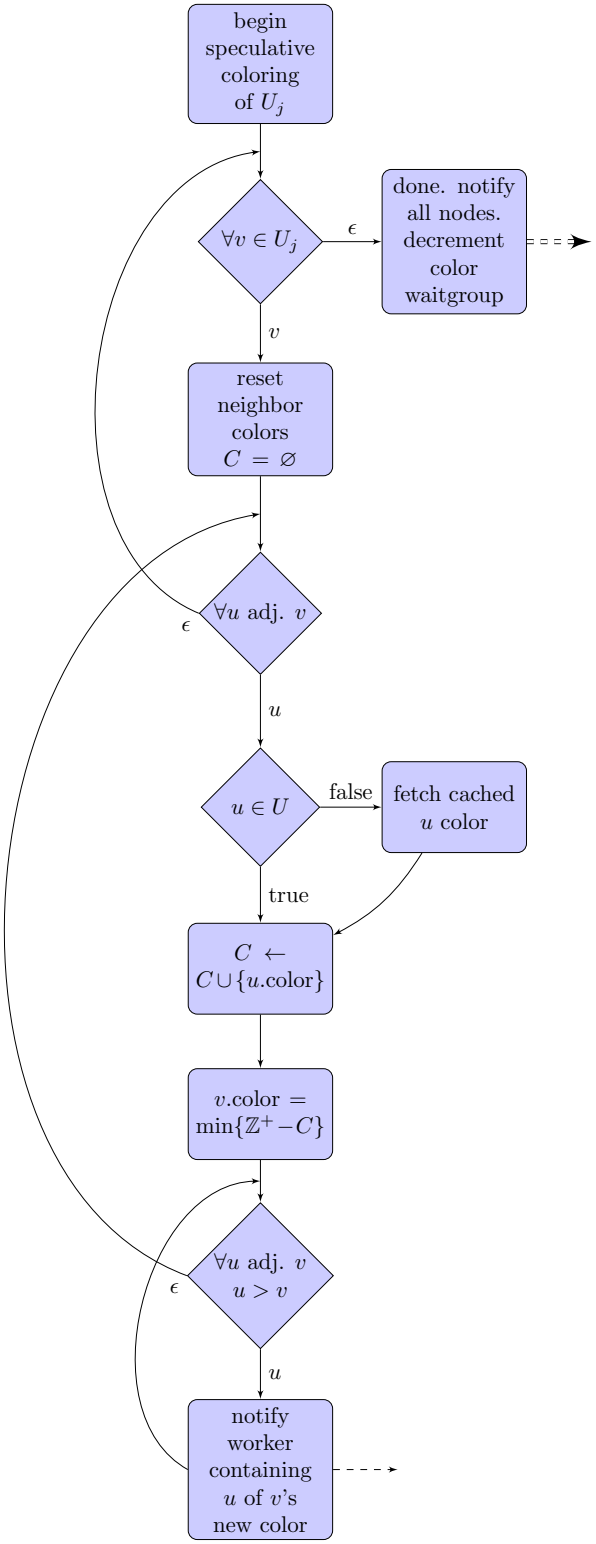
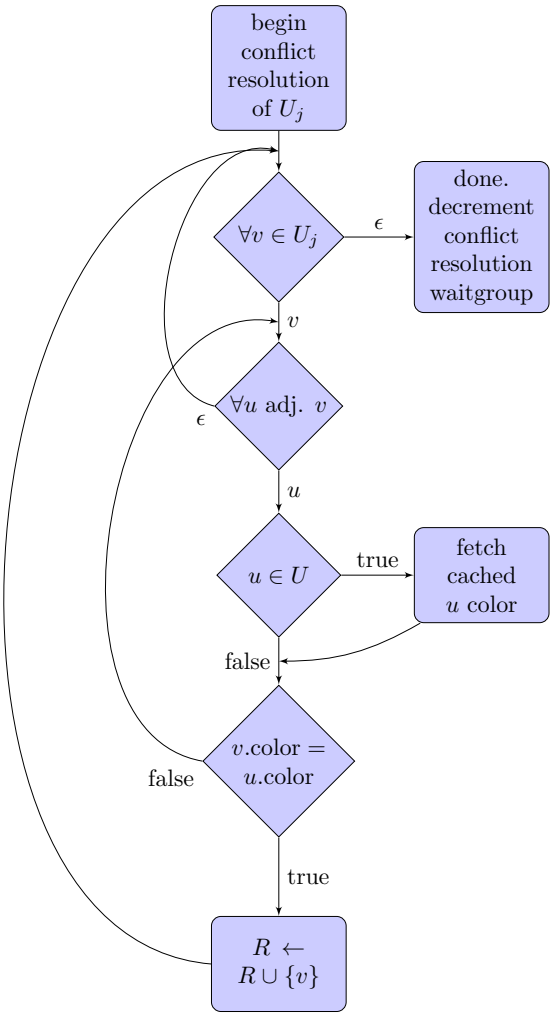


Figure 3: Detail of the "distributed coloring algorithm" block from Figure 1 for a single worker node, based on the general framework proposed by Gebremedhin et al. (2005). The double-dashed arrow represents the set of connections to all of the other nodes. The blocks that involve parallel processing are indicated by a double border. The listener (the same listener set up during the procedure) is in a separate goroutine and therefore will not block when the mainline is blocking on a semaphore.



(a) The speculative coloring algorithm for a single thread



(b) The conflict detection algorithm for a single thread

Figure 4: Detail of the speculative coloring and conflict detection algorithms in Figure 3. Note that the graph is first partitioned amongst the worker nodes $G = \bigcup G_i$. For a particular worker, U represents the nodes to be colored and is set to G_i initially. It is then partitioned again into $U = \bigcup U_j$, and thus we achieve a fairly high level of parallelization. This part of the algorithm is not much different from the single-node variant, except that we require fetching cached color values of neighbors that lie outside the current subgraph, and we have to notify worker nodes containing neighbors of nodes that have been recolored. To reduce network I/O, we arbitrarily only notify the higher-indexed neighbors when a node is colored (hence the $u > v$ condition).

3 Fixing concurrency issues

3.1 Short writes

This was a tricky issue for a number of reasons:

- It's not a common issue (it doesn't appear anywhere on an Internet search)
- The write buffer was often not full.
- The listener (reader) thread doesn't hang.
- The typical network socket buffer is large (16kB, according to `man 7 tcp`), and substantially larger than the default buffer size for `bufio.Writer` (4kB).

The only thing that did make sense was that a short write is a reasonable (?) response for a FIFO like a network socket or a pipe, but it must not be common on ordinary network sockets because an Internet issue doesn't turn it up.

Painstaking debugging revealed that the short writes were really very random and the buffer was not full at all, and it was due to concurrent calls to the `Write()` method of the underlying socket which returned 0. This is solved by making all writes to any particular network socket a critical section.

(Debugging first revealed the fact that a short write causes a `bufio.Writer` to reject *all future writes* and maintain its error until a call to the writer's `Reset()` method is made, which also discards the buffer. Only when messing around with this did the real culprit come up.)

3.2 Synchronizing the start of a coloring round

The code in Figure 5 shows the general distributed algorithm sequence, along with the synchronization steps. The problem with this code is that `colorWg` may be decremented before it is set. A possible series of events, assuming a two-worker system is: the first node (node1) begins a round, and broadcasts `ROUND_SYNC`, and then it sleeps on `startWg.Wait()`. The second node (node2) begins a round, broadcasts `ROUND_SYNC`, receives node1's `ROUND_SYNC` message (releasing `startWg`), and doesn't sleep on `startWg.Wait()` since the semaphore is released. It (node2) continues on its mainline thread: it sets `startWg` and `colorWg` to their appropriate values, performs its speculative coloring, broadcasts `SPEC_COLOR_DONE`, and sleeps on `colorWg.Wait()`. Then, node1's listener thread receives the `ROUND_SYNC` message, freeing the `startWg` semaphore (but this doesn't cause the mainline thread to immediately wake/get scheduled in), and it receives the `SPEC_COLOR_DONE` message, decrementing the `colorWg` semaphore to -1. This is due to the fact that node1's mainline thread still hasn't woken yet and set the `colorWg` semaphore appropriately, and the negative semaphore raises a panic.

```
1  colorWg.Add(0)
2  startWg.Add(nNodes - 2)
3  U := G_i
4  while U is not empty:
5
6      // round sync
7      broadcast ROUND_SYNC message
8      startWg.Wait()
9
10     // these have to happen after
11     // startWg because we need to know
12     // how many nodes are left
13     startWg.Add(nNodes - 2)
14     colorWg.Add(nNodes - 2)
15
16     // perform speculative coloring ...
17     speculativeColor()
18     broadcast SPEC_COLOR_DONE message
19     colorWg.Wait()
20
21     // conflict detection
22     R := []
23     conflictDetection()
24     U = R
25
26 // node completely done
27 broadcast NODE_DONE message
```

(a) Mainline thread

```
1  function handler_ROUND_SYNC() {
2      startWg.Done()
3  }
4
5  function handler_NODE_DONE() {
6      nNodes = nNodes - 1
7      startWg.Done()
8  }
9
10 function handler_SPEC_COLOR_DONE() {
11     colorWg.Done()
12 }
```

(b) Listener thread

Figure 5: A seemingly innocuous bit of synchronization that caused a major headache. Note that the number of nodes may decrease throughout the lifetime of the algorithm (due to nodes completing before other nodes), so it is important to count the number of `ROUND_SYNC` and `NODE_DONE` messages before setting the semaphores for the round; thus lines 13 and 14 explicitly follow the `startWg`. However, this causes problems: after `startWg` is released, the mainline thread may not get scheduled in immediately; it is possible that the listener thread may receive the `SPEC_COLOR_DONE` message and try to decrement `colorWg` before it is set, which causes a negative WaitGroup error.

One proposed way around this is to set `colorWg` before `startWg` is released, i.e., that `ROUND_SYNC` should increment `colorWg` in its handler, and this replaces line 14 of the mainline. However, this causes the possible race condition: assuming a two-worker system, node1 can be sleeping on `colorWg.Wait()`. Node2 then finishes speculative coloring, releasing `colorWg`, but node1's mainline thread does not get scheduled in immediately. Node2 continues to run, finishes the round, begins the next round, and broadcasts the `ROUND_SYNC` message. Node1's listener thread receives the message and decrements `colorWg` (per the new change) to a value of 1. However, since the mainline thread hasn't

woken yet, we are greeted with a error of "sync: WaitGroup is reused before previous Wait has returned."

A second proposed method to get around this is to perform locking of critical sections (e.g., using mutexes) to try eradicate the mistiming issues. However, no suitable way to do this was found that would block all mistimed WaitGroup operations and not also deadlock the main thread.

A third proposed method was to use a series of ACK messages (similar to TCP ACKs) to signal when a node is ready to accept WaitGroup operations. However, this adds an additional layer of synchronization that slows down the algorithm further and introduces an additional message type and handler, which is a lot of complexity (and thus a decrease in maintainability).

The final (working) solution is based on the idea that the fault in the original algorithm is simply due to a scheduling race condition. This solution involves trying to decrement `colorWg` in the handler for `SPEC_COLOR_DONE`; if this operation fails, then we catch the panic (exception) and force Go to yield this thread to the scheduler. This process is repeated as many times as necessary until the operation succeeds, at which point we know that the main thread has updated `colorWg`'s counter.

This method is a little tricky, since the only way to catch a panic is by a `defer`red handler on the stack; this defer method has to be recursive but have the proper break condition. The other tricky part is that when `colorWg.Done()` causes the negative WaitGroup panic, it still decrements the pointer; so, to ensure consistency, `colorWg` must be (atomically) re-incremented to zero, or else there might be problems on the mainline thread when it tries to correctly increment `colorWg`. Thus this introduces a critical section around the decrementing of `colorWg` and around the setting of `colorWg` in the mainline thread.

The last step is to schedule out the offending thread if it panics (i.e., if it runs too early). `runtime.Gosched()` does exactly this.

This last method is perhaps not great because it doesn't avoid the error, but instead it provides a sort of atomic "test and set" (TAS)-like functionality by using a lock and a semaphore. The benefit is that it only requires an extra mutex (one per socket connection) and, while it forces rescheduling (usually an expensive operation), this is only the Goroutine scheduler (more lightweight than the OS scheduler) and forces the correct (desired) scheduling order.

```
1  // recursive error recovery "trick" -- see documentation
2  var retry func()
3  retry = func() {
4      // consume error; keep retrying until success
5      if err := recover(); err != nil {
6
7          // if this happens, then lock already acquired
8          ws.ColorWg.Add(1)
9          ws.ColorWgLock.Unlock()
10
11         // algorithm is done, don't need to continue listening
12         if ws.State == distributed.STATE_FINISHED {
13             return
14         }
15
16         defer retry()
17
18         // give control to someone else (hopefully the main worker
19         // thread)
20         runtime.Gosched()
21
22         // retry
23         ws.ColorWgLock.Lock()
24         ws.ColorWg.Done()
25         ws.ColorWgLock.Unlock()
26     }
27 }
28 defer retry()
29
30 logger.Printf("Node %d has finished a round.\n", nodeIndex[0])
31
32 // decrease the number of nodes we are waiting for
33 ws.ColorWgLock.Lock()
34 ws.ColorWg.Done()
35 ws.ColorWgLock.Unlock()
```

Figure 6: The fix to the synchronization problem. Note that this also requires acquiring the `ws.ColorWgLock` for line 14 of 5 (and releasing it thereafter).

4 Performance issues

Mostly due to a lack of time, we haven't been able to improve performance by much. A 10000 vertex graph with an average degree of 1000, on a server and four virtual clients running on the same machine, takes almost a minute to color using the distributed algorithm, while it takes roughly 17ms using the single-node, multi-threaded algorithm. This is due to the following reasons:

4.1 Unavoidable overhead from distributed algorithm

The biggest drawback of this algorithm is that the graph is now distributed across the RAM of multiple nodes, meaning that efficient access to a node's neighbors (which is critical for this coloring algorithm) is much, much slower. As a result, a good partitioning is highly important for performance.

There is also the overhead of requiring each node to be synchronized at the start of each round, which waste CPU time. Logging also adds overhead. Running all the clients on the same host probably causes a slowdown as well due to sharing the same network interface.

4.2 Graph generation and partitioning

Real-world graphs tend to be very diverse in terms of vertex degree, and generally are heavily skewed with a large proportion of nodes having very small degree. This also means that they can be efficiently partitioned in a way that minimizes cross-edges between subgraphs. Gebremedhin et al. (2005) used real-world graphs from a disease dataset and partitioned using the METIS graph partitioning package, while our graphs were generated with a uniform degree and partitioned into linear blocks. In other words, we performed no optimization yet on minimizing internode I/O, and this means that much of our algorithm time is probably stuck in networking time.

4.3 Increased vertex info buffering

Currently, due to the lack of an extra data structure to store multiple vertices, vertices are sent and received one at a time. I/O speed can probably be improved by sending and receiving in larger chunks.

5 Next steps

The next step is to move this onto a truly distributed system (multiple physical/virtual hosts rather than the same physical and virtual host) using AWS infrastructure-as-code. This will provide the challenge of working with more numerous but less-individually-powerful nodes (standard T2 instances are single-threaded and are RAM-limited).

Naïve Bayes Text Categorizer

Jonathan Lam

March 19, 2021

The code is available on GitHub at jlam55555/naive-bayes-text-categorization.

1 Build instructions

The build instructions can be found on the README on GitHub.

2 About the algorithm

2.1 Tokenization

Tokenization is first performed using nltk's `word_tokenizer`. No filtering (e.g., removing stop words or punctuation) is performed on tokens.

2.2 General algorithm

Naïve Bayes works by assuming that each word is conditionally independent of each other word given its class, which allows us to express the probability of a document D belonging to class C as:

$$P(C \mid D) = P(C) \prod_i P(w_i \mid C)$$

where $w_i \in D$ is a token in D. We can replace the product by a sum if we take the logarithm of the probabilities, and take the argmax to find the most probable class, i.e.:

$$\hat{C}_D = \operatorname*{argmax}_C \log P(C) \sum_i \log P(w_i \mid C)$$

2.3 General implementation

2.3.1 Attempt 1

A fairly basic Naïve Bayes algorithm is used for training and categorization. First, a list of all of the vocabulary words and categories is collected. A numpy

matrix of size $V \times C$ is created to store term-class counts (to calculate class-conditional probabilities), where V is the size of the vocabulary and C is the number of classes. Another numpy vector of size $1 \times C$ is created to store the document frequency for each class (to calculate class priors).

During training, term and document counts are aggregated in the two matrices described above. When training is complete, the $V \times C$ matrix is converted to log-class-conditional probabilities by dividing the term counts by the total term frequency of the class. I.e. for a class C and term w_i:

$$P(w_i \mid C) = \frac{P(w_i \cap C)}{P(C)} = \frac{P(w_i \cap C)}{\sum_j P(w_j \cap C)} = \frac{\text{count}(w_i \cap C)}{\sum_j \text{count}(w_j \cap C)}$$

The log-priors are calculated by taking the logarithm of the frequency of documents having class C:

$$P(C) = \frac{\text{count(documents in class } C)}{\text{count(documents)}}$$

And thus we have all the components we need for the Naïve Bayes calculation. Calculating sums, dividing by total counts, and taking the elementwise logarithm was easy using numpy's vectorized array operations.

When performing inference, words not in the training vocabulary are simply discarded.

Laplace smoothing was used to prevent zero probabilities. This was chosen because it is simple and is known to work fairly well. No more complicated smoothing schemes were attempted, and this is a room for future improvement. For the first attempt, the smoothing factor was 1 (i.e., a +1 smoothing).

2.3.2 Attempt 2

To improve the speed of the program, the algorithm switched to using dictionaries rather than a dense numpy array. This had a (qualitatively) small speedup on the algorithm. This in itself should not change the performance (accuracy) of the algorithm.

I experimented with the smoothing factor. It seemed that a smoothing factor between 0.05 to 0.07 worked best, empirically, although this could be refined further through some automated hyperparameter search (which was not attempted for lack of time).

I also experimented with lemmatization, stemming, and stopword removal. Removing stopwords (using `nltk.corpus.stopwords`) had no considerable effect on the accuracy, but lemmatization and stemming had a noticeable effect. Notably, stemming performed slightly better than lemmatization, and either the porter or the snowball stemmers seemed to work well.

2.3.3 Attempt 3

I tried a different formula for the conditional probabilities following the recommendation from lecture. This comes from a slightly different interpretation and

reasoning behind the conditional probabilities:

$$P(w_i \mid C) = \frac{\text{count(documents in class } C \text{ containing } w_i)}{\text{count(documents in class } C)}$$

the new interpretation being that this was the chance that a document contains a word (at least once).

However, this method did not achieve better results. While the results were similar for corpora 1 and 2, the accuracy significantly dropped for corpus 3 for unknown reasons. (It strongly favors the USN category.) This method should offer higher accuracy, so it is unknown what is working correctly.

3 Performance

3.1 Evaluating performance

The file `train_test_split.py` partitions the training datasets into a training and validation (evaluation) dataset with a default 80/20 train/test split. The performance metrics are then calculated using the provided `analyze.pl` script.

The following results are shown using the provided train/test split for corpus 1, as well as typical random 80/20 train/test split results for corpus2 and corpus3 (these particular splits and results are stored on the GitHub repo as well).

3.2 Attempt 1

3.2.1 Corpus 1

```
Processing answer file...
Found 5 categories: Pol Cri Oth Dis Str
Processing prediction file...

381 CORRECT, 62 INCORRECT, RATIO = 0.860045146726862.

CONTINGENCY TABLE:
        Pol     Cri     Oth     Dis     Str     PREC
Pol     122     2       11      0       5       0.87
Cri     0       38      0       0       1       0.97
Oth     1       0       6       0       0       0.86
Dis     0       0       5       86      0       0.95
Str     21      10      3       3       129     0.78
RECALL  0.85    0.76    0.24    0.97    0.96

F_1(Pol) = 0.859154929577465
F_1(Cri) = 0.853932584269663
F_1(Oth) = 0.375
F_1(Dis) = 0.955555555555556
F_1(Str) = 0.857142857142857
```

3.2.2 Corpus 2

```
Processing answer file...
Found 2 categories: O I
Processing prediction file...

155 CORRECT, 24 INCORRECT, RATIO = 0.865921787709497.

CONTINGENCY TABLE:
        O       I       PREC
O       117     14      0.89
I       10      38      0.79
RECALL  0.92    0.73

F_1(O) = 0.906976744186046
F_1(I) = 0.76
```

3.2.3 Corpus 3

```
Processing answer file...
Found 6 categories: Sci Fin Wor USN Spo Ent
Processing prediction file...

150 CORRECT, 25 INCORRECT, RATIO = 0.857142857142857.

CONTINGENCY TABLE:
        Sci     Fin     Wor     USN     Spo     Ent     PREC
Sci     20      0       0       0       0       0       1.00
Fin     1       19      0       0       0       2       0.86
Wor     0       0       53      3       1       1       0.91
USN     5       2       3       43      3       4       0.72
Spo     0       0       0       0       14      0       1.00
Ent     0       0       0       0       0       1       1.00
RECALL  0.77    0.90    0.95    0.93    0.78    0.12

F_1(Sci) = 0.869565217391304
F_1(Fin) = 0.883720930232558
F_1(Wor) = 0.929824561403509
F_1(USN) = 0.811320754716981
F_1(Spo) = 0.875
F_1(Ent) = 0.222222222222222
```

3.3 Attempt 2

3.3.1 Corpus 1

```
Processing answer file...
Found 5 categories: Cri Dis Str Oth Pol
Processing prediction file...

393 CORRECT, 50 INCORRECT, RATIO = 0.887133182844244.

CONTINGENCY TABLE:
Cri     Dis     Str     Oth     Pol     PREC
```

```
Cri     40      0       2       1       1       0.91
Dis     0       89      1       3       0       0.96
Str     8       0       128     3       18      0.82
Oth     0       0       0       13      2       0.87
Pol     2       0       4       5       123     0.92
RECALL  0.80    1.00    0.95    0.52    0.85

F_1(Cri) = 0.851063829787234
F_1(Dis) = 0.978021978021978
F_1(Str) = 0.876712328767123
F_1(Oth) = 0.65
F_1(Pol) = 0.884892086330935
```

3.3.2 Corpus 2

```
Processing answer file...
Found 2 categories: I O
Processing prediction file...

159 CORRECT, 20 INCORRECT, RATIO = 0.888268156424581.

CONTINGENCY TABLE:
I       O       PREC
I       38      6       0.86
O       14      121     0.90
RECALL  0.73    0.95

F_1(I) = 0.791666666666667
F_1(O) = 0.923664122137405
```

3.3.3 Corpus 3

```
Processing answer file...
Found 6 categories: Fin Ent Sci USN Spo Wor
Processing prediction file...

157 CORRECT, 18 INCORRECT, RATIO = 0.897142857142857.

CONTINGENCY TABLE:
Fin     Ent     Sci     USN     Spo     Wor     PREC
Fin     20      2       3       0       0       0       0.80
Ent     0       4       0       0       0       0       1.00
Sci     0       0       21      0       0       1       0.95
USN     1       1       2       45      2       3       0.83
Spo     0       0       0       0       15      0       1.00
Wor     0       1       0       1       1       52      0.95
RECALL  0.95    0.50    0.81    0.98    0.83    0.93

F_1(Fin) = 0.869565217391304
F_1(Ent) = 0.666666666666667
F_1(Sci) = 0.875
F_1(USN) = 0.9
F_1(Spo) = 0.909090909090909
F_1(Wor) = 0.936936936936937
```

3.4 Attempt 3

3.4.1 Corpus 1

```
Processing answer file...
Found 5 categories: Oth Dis Pol Str Cri
Processing prediction file...

390 CORRECT, 53 INCORRECT, RATIO = 0.880361173814898.

CONTINGENCY TABLE:
Oth     Dis     Pol     Str     Cri     PREC
Oth     9       0       2       0       0       0.82
Dis     4       89      0       1       0       0.95
Pol     9       0       130     10      1       0.87
Str     3       0       12      123     10      0.83
Cri     0       0       0       1       39      0.97
RECALL  0.36    1.00    0.90    0.91    0.78

F_1(Oth) = 0.5
F_1(Dis) = 0.972677595628415
F_1(Pol) = 0.884353741496599
F_1(Str) = 0.869257950530035
F_1(Cri) = 0.866666666666667
```

3.4.2 Corpus 2

```
Processing answer file...
Found 2 categories: O I
Processing prediction file...

156 CORRECT, 23 INCORRECT, RATIO = 0.871508379888268.

CONTINGENCY TABLE:
O       I       PREC
O       118     14      0.89
I       9       38      0.81
RECALL  0.93    0.73

F_1(O) = 0.911196911196911
F_1(I) = 0.767676767676768
```

3.4.3 Corpus 3

```
Processing answer file...
Found 6 categories: USN Ent Spo Wor Sci Fin
Processing prediction file...

139 CORRECT, 36 INCORRECT, RATIO = 0.794285714285714.

CONTINGENCY TABLE:
USN     Ent     Spo     Wor     Sci     Fin     PREC
USN     44      3       5       5       11      7       0.59
Ent     0       2       0       0       0       0       1.00
```

```
Spo      0       0       13      0       0       0       1.00
Wor      2       1       0       51      0       0       0.94
Sci      0       0       0       0       15      0       1.00
Fin      0       2       0       0       0       14      0.88
RECALL   0.96    0.25    0.72    0.91    0.58    0.67

F_1(USN) = 0.727272727272727
F_1(Ent) = 0.4
F_1(Spo) = 0.838709677419355
F_1(Wor) = 0.927272727272727
F_1(Sci) = 0.731707317073171
F_1(Fin) = 0.756756756756757
```

3.5 Summary

Attempt	Corpus 1	Corpus 2	Corpus 3	Average
1	86.0	86.6	85.7	86.1
2	88.7	88.8	89.7	89.1
3	88.0	87.1	79.4	84.8

Table 1: Summary of performances of all attempts

3.6 Speed / Efficiency

The algorithm is not very fast, taking 10-20 seconds on the first and third corpus.

There is the potential for extreme speedup through parallelization, since the training stages can be entirely parallelized between train or evaluation cases (but the training and evaluation stages cannot be overlapped).

ECE467 – Pset 1

Jonathan Lam

February 27, 2021

1. Consider the following sentence: "*Engineering is a complex endeavor.*" Assume that an HMM POS tagger is used to choose between these two tag-sequences (involving Penn Treebank POS tags) for the given sentence:

   (a) `Engineering/NN is/VBZ a/DT complex/NN endeavor/NN ./.`

   (b) `Engineering/NN is/VBZ a/DT complex/JJ endeavor/NN ./.`

   In other words, the system is only considering a single tag for most of the words, but the word "complex" can be tagged as either a singular noun or an adjective. Without stating any actual numbers, show which tag transition probabilities and word observation likelihoods (a.k.a. emission probabilities) will determine the choice for this tag. In terms of these probabilities, express an equation that must hold for the JJ tag to be chosen.

   In general, an HMM tries to fit the tag sequences following the equation (assuming a bigram approximation):

$$\hat{t}_1^n \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} [P(w_i \mid t_i) P(t_i \mid t_{i-1})]$$

   JJ would be chosen as the tag for "complex" if:

$$P(\text{"complex"} \mid \text{JJ}) P(\text{JJ} \mid \text{DT}) P(\text{NN} \mid \text{JJ}) > P(\text{"complex"} \mid \text{NN}) P(\text{NN} \mid \text{DT}) P(\text{NN} \mid \text{NN})$$

   where $P(\text{"complex"} \mid \text{JJ})$ represents an emission probability and $P(\text{JJ} \mid \text{DT})$ represents a transition probability. (These are the only terms that are different in the approximation.)

2. Consider all words representing the sound a cow makes to be the following infinite set: "moo!", "mooo!", "moooo!", etc. Show a regular expression that would accept any line in which the sound a cow makes occurs at least two times (they don't have to be consecutive or identical). Each instance must be preceded by a non-letter or occur at the start of a line. It does not matter what occurs after the explanation points.

   Using standard PCRE syntax, with non-capturing groups:

   ```
   (?:(?:^|[^a-zA-Z])moo+!.*){2,}
   ```

3. Assume a bigram model and a trigram model are both trained on a large corpus of English that includes many technical, AI-oriented documents. Consider the bigram probability associated with the phrase "language processing" and the trigram probability associated with the phrase "natural language processing". Which do you expect would be larger? Explain your answer.

   The bigram probability is $P(\text{"processing"} \mid \text{"language"})$. The trigram probability is $P(\text{"processing"} \mid \text{"natural"}, \text{"language"})$. The trigram probability is likely higher because when you say "natural language" you are typically talking in a scientific context and are likely talking about NLP (especially in an AI-related corpus), whereas the word "language" can be followed by many different words and is more likely not related to a phrase on language processing.

4. What is the major linguistic distinction between languages such as Mohawk and languages such as English?

   Mohawk is a "polysynthetic language," which means that "verbs must include some expression of each of the main participants in the event described by the verb (the subject, object, and indirect object)." English does not have this parameter.

5. Consider an information retrieval system that relies on a vector space model and TF*IDF weights. A straight-forward system would have to loop through all documents in the collection to compute the similarity between the query and each document. Briefly explain how an inverted index can be used to make the IR system much more efficient.

   A full document-term matrix would be very large and sparse. An inverted-index data structure would map each term to which documents it occurs in and how many times it occurs in those documents, which is sufficient to calculate the TF*IDF metric while saving space and only looping though documents that contain the term.

ECE467 – Pset 2

Jonathan Lam

April 6, 2021

1. Consider the context-free grammar (CFG) shown in Figure 12.3 on page 235 of the current draft of the textbook, and also shown on slide 9 of my PowerPoint presentation on Phase Structure Grammars and Dependency Grammars. Figure 12.2 shows a related lexicon. Now consider the following sentence, which is valid according to the grammar and lexicon, ignoring case and punctuation: "You need a flight to Chicago". Draw a valid parse tree for this sentence, according to the grammar.

```
[S
        [NP [Pronoun you]]
        [VP
                [Verb need]
                [NP
                        [Det a]
                        [Nominal [Noun flight]]]
                [PP
                        [Preposition to]
                        [NP [Proper-Noun Chicago]]]]]
```

2. Briefly explain the difference between a prescriptive grammar and a descriptive grammar. Which do most modern linguists strive to develop?

   A prescriptive grammar attempts to formalize a language into a series of rules that all the syntax follows. While this is a cleaner approach and may be applied to more formal grammars, it is hard to make everything conform to a small set of rules in a natural language. Therefore most modern linguists strive to develop a descriptive grammar, which attempts to describe how the language is actually used (e.g., how a native speaker would use the language), including all of its pecularities. Descriptive grammars do not attempt to make everything conform to these rules.

3. Consider a CFG representing a phrase structure grammar for a natural language such as English, including the following rule (the period is part of the rule): S → NP VP.

   In this rule, S, NP, and VP are non-terminal symbols, but we will assume that the period is a terminal symbol specifically representing a period (as opposed to some other end-of-sentence marker). So, we can interpret this rule as stating that a sentence can have the form of a noun phrase followed by a verb phrase followed by a period. (To implement a parser for grammars with rules like this, the tokenizer would have to treat periods as separate tokens.)

   Assume that you need to convert this grammar to Chomsky normal form (CNF), perhaps with the CKY parsing algorithm in mind. Using the procedure discussed in class, show the rules that might result from the conversion.

   Two possible CNF grammars:

   (a) S → _Dummy1 Period
   Dummy1 → NP VP
   Period → .

   (b) S → NP _Dummy2
   _Dummy2 → VP Period
   Period → .

4. Explain any serious shortcoming of first-order logic as a meaning representation.

   It only handles things in absolutes: an element is in a set, or it is not; it is related to another element, or it is not. There is no part of the formulation to account for uncertainty.

5. Invent your own Winograd schema! Circle the ambiguous pronoun in both sentences, and underline the correct antecedent in each.

   The ambiguous pronoun is "she," which could be referring to "my sister" or "our dog." The ambiguity is resolved by the verb "wagged" or "pet."

   My sister came to see **our dog**. **She** *wagged* her tail.

   My sister came to see our dog. **She** *pet* her tail.

ECE467 – Pset 3

Jonathan Lam

May 10, 2021

1. Consider a neural language model (NLM) that uses a softmax layer as its output layer. How many nodes where there be in the output layer? What is the typical interpretation of the value of each output node?

   NLM is somewhat of a categorization problem to determine the most probable next word. Thus, there would be as many nodes as the size of the vocabulary, with the interpretation of the output being a probability distribution over how likely it is that each word comes next.

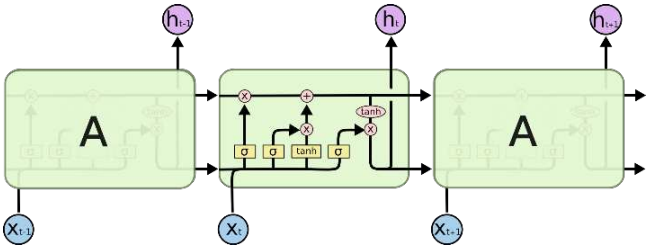
2. Assume that EMB(w) is the word embedding for a word, w, computed by some common technique for computing word embeddings (e.g., word2vec). Consider the vector obtained by the following vector arithmetic:

   EMB("Japan") + EMB("Paris") - EMB("Tokyo")

   Other than the words that already related to this equation, what word's embedding would likely be close to the result? Briefly explain (in one sentence) your answer.

   Hopefully, "France" should be close: the intuition is that vector embeddings hold semantic information in their numeric components, so we can apply "vector math" to analogies. (We can think of this, very loosely, as starting with the country Japan, adding the country France and the idea of a capital, and then subtracting out the idea of Japan and its capital; this leaves us with the France component.)

3. Consider the enrolled depiction of a few LSTM cells below, which was discussed in class and was taken from https://colah.github.io/posts/2015-08-Understanding-LSTMs. What is represented by the output of each of the three sigmoid functions in the cell (please limit your responses to one sentence each).



The sigmoid functions each act as a gate to decide what information to let through. From left to right, the first is the "forget gate," which decides which parts of the context to throw away; the second is the "input gate," which decides which parts of the context to update; and the third is the "output gate," which decides what to emit to the cell's hidden state.

4. We have learned that encoder-decoder networks (a.k.a. sequence-to-sequence models) can be useful for certain NLP tasks, such as machine translation (MT). For MT, the encoder produces a context vector, which can be thought of as representing the input sentence from a source language, while the decoder generates a predicted translation in the target language. Briefly explain (at a high level, using one or two sentences) why adding attention to such a model can be useful.

In seq2seq models where there isn't a one-to-one mapping from input to output (nor is the mapping usually in order, as different languages tend to have very different sentence structure), we need a way to choose a (fixed-length) context vector that is representative of the meaning of the sentence. Training the decoder with attention allows it to use a more complex function of the hidden states that may be able to produce more fluent translations (in the case of MT) or other decodings.

5. Briefly explain (in one sentence) one major reason why producing embeddings for characters or subwords instead of, or in addition to, full words can be useful.

It may often be unrealistic to learn embeddings for each word in a language, because there are simply too many (and some words may be extremely rare) but morphological parts of a word (subwords) tend to be much more common; this does well on out-of-vocabulary prediction.

ECE467 Final Project – Sentiment Analysis

Jonathan Lam, Derek Lee, Victor Zhang

May 11, 2021

1 Introduction

We are doing a sentiment analysis task on Twitter tweets. This was inspired by the IEEE 2021 Global Student Challenge[1], in which one challenge is to "analyz[e] sentiments in tweets related to the COVID-19 pandemic."

We chose to use the TensorFlow BERT uncased pre-trained model to perform regression on a numerical sentiment value and classification on classes created by binning those sentiment values. Our model is trained and evaluated on the IEEE "Coronavirus (COVID-19) Geo-Tagged Tweets" dataset[2].

2 Implementation Details

Our code can be found on GitHub at @jlam5555/nlp-sentiment-analysis.

2.1 Dataset

We chose an arbitrary dataset containing tweets related to COVID-19. From the abstract pertaining to this particular dataset:

> This dataset contains IDs and sentiment scores of geo-tagged tweets related to the COVID-19 pandemic. The real-time Twitter feed is monitored for coronavirus-related tweets using 90+ different keywords and hashtags that are commonly used while referencing the pandemic. Complying with Twitter's content redistribution policy, only the tweet IDs are shared.

This dataset contains approximately 360,000 tweets. We did not end up using the geotags, instead focusing only on the tweet contents. (This is actually a subset of a larger dataset[3] containing over 1.2 billion tweets relating to COVID-19, but we only discovered this later, and we figured that the geo-tagged tweets dataset is large enough for our purposes.)

[1] `https://www.computer.org/publications/tech-news/events/global-student-challenge-competition-2021`

[2] `https://ieee-dataport.org/open-access/coronavirus-covid-19-geo-tagged-tweets-dataset`

[3] `https://ieee-dataport.org/open-access/coronavirus-covid-19-tweets-dataset`

As mentioned in the abstract, only the tweet IDs and sentiment labels were provided. The tweets were grouped by day, but we ignored this (only taking into account tweet contents). The tweet contents were fetched using the tweet IDs and the Hydrator tool[4], which uses the Twitter API v2.0 (which requires a Twitter developer account). This returned a JSON file containing all of the approximately 270,000 available (non-hidden and non-deleted) tweets as of April 25th, which forms our main dataset.

The main dataset is this dataset combined with the original labels, with all of the extra metadata trimmed.

2.2 Preprocessing

The ALBERT (A Lite BERT) tokenizer is used for tokenization:

```
input_segments = [
    tf.keras.layers.Input(shape=(), dtype=tf.string, name=ft)
    for ft in sentence_features]

# Tokenize the text to word pieces.
bert_preprocess = hub.load("http://tfhub.dev/tensorflow/albert_en_preprocess/3")
tokenizer = hub.KerasLayer(bert_preprocess.tokenize, name='tokenizer')
segments = [tokenizer(s) for s in input_segments]
```

2.3 Model

The inputs are the hydrated tweet strings (up to 280 characters). We are given sentiment labels, which measure how positive the sentiment of a tweet is. Our main model was composed of an ALBERT tokenizer followed by a BERT encoder.

We performed regression and classification on our dataset. For both of the tasks, we used the main model and appended a few layers on top of it.

For the regression model, we appended a Dropout layer, a BatchNorm layer, and 4 dense layers. For the classification model, we appended a BatchNorm layer, 5 dense layers, and a softmax layer.

We used an 80/20 train/test split. We did not do any hyperparameter tuning, so the test set was also used as the validation set to view the model's performance over the course of training.

2.3.1 Experimentation with network topologies

For regression, our original network comprised 3 dense layers with a single output. Each dense layer used a ReLU activation. The first layer had a width of 128 nodes, the second 64 nodes, and the third 32 nodes (the final layer had 1 node representing the output).

[4] https://github.com/DocNow/hydrator

When training, we realized that the result had almost no variance in the output, so the main goal following this was to try to improve the expressibility of the model's output. Almost all of the values outputted by the model were very close to the mean sentiment value (around 0.11).

We made several attempts to improve the model's expressibility (all to no avail), including:

- Experimenting with the learning rate. One conjecture was that the model wasn't learning fast enough or was converging too quickly, so we tried several builtin optimizers (Adam, SGD), learning rates (the default is 0.001, so we tried 0.01 and 0.1), and learning rate scheduling (e.g., high learning rate at the beginning followed by a lower learning rate, as suggested by this paper).

- The encodings from the BERT encoder may not have enough variance – this may be true if the tweets are considered fairly similar by BERT. We tried using a BatchNorm layer to combat this.

- The model may simply not be large enough. We discovered that the BERT encoding is a 768-length encoding, so even our largest first layer was already constricting the network width. We tried expanding the width and the depth of the layer, trying as wide and deep as a $1024 \times 512 \times 256 \times 128 \times 64 \times 1$ network.

- After examining the distribution of the sentiment labels, we noticed that a very large number of the examples were exactly zero, which likely caused the model to train to roughly a constant value. We tried to combat this by binning the labels into three roughly equal-sized categories and turning the problem into a classification task. However, our results stayed at the baseline levels (i.e., as if we had chosen to most likely category).

- Played around with different activations (e.g., PReLU, ELU), initializers (glorot_uniform, glorot_normal, he_uniform, he_normal) in our desperation.

For classification, our network comprised 4 dense layers with 3 output nodes. Each dense layer used a ReLU activation. The first layer had a width of 512 nodes, the second 256, the third 128, the fourth 64 (the final layer had 3 nodes representing each of the 3 classes).

When training, we encountered similar problems when we were training the regression model. We made similar attempts to improve the model, but the outputs of our model were completely uniform, classifying every input as the most frequent class.

2.3.2 Other challenges

Training time is very slow with the BERT preprocessing and encoder layers. Each training epoch takes roughly 1.5 hours, which means that we were not

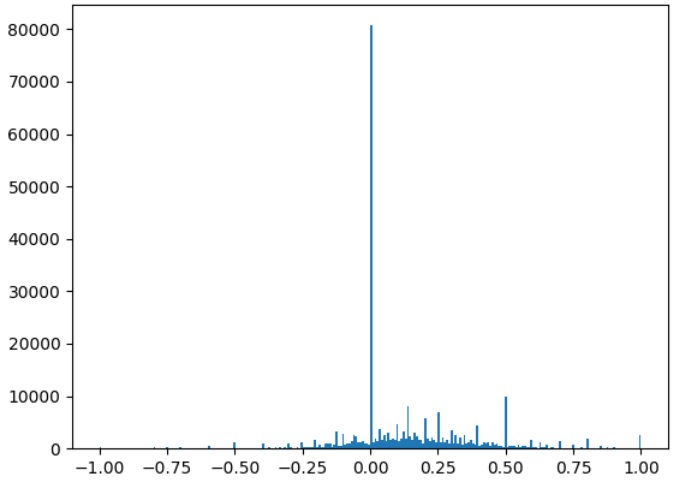


Figure 1: Distribution of the labels. The sentiment labels are on the x-axis. The frequency is on the y-axis. We decided to try binning the labels in order to increase the "variance" of the output.

able to experiment with different models very quickly. Even when all of us were training in parallel, it was hard to make much progress.

Another issue is that, just due to the complexity of TensorFlow, it's difficult to meaningfully "debug" the network. We speculate that our inputs may not be well-suited for the BERT model – even though large neural language models tend to work well on many NLP tasks, our dataset is fairly application-specific. The distribution of the English language in tweets is probably very different than that of the general English language. For example, many of the tweets contained Unicode emojis, which may not have been tokenized properly by the tokenizer.

3 Results

For both the regression and categorization tasks, the best results we got were very close to the baseline. For regression, the best results were similar to the sample mean. For classification, the best results were similar to the accuracy if we tried to classify all inputs as the most frequent class. In other words, our model trained to predict the average label with very little variance, which (probably) indicates that there was a problem with the output of the BERT encoder. (This is the same result that one would get if random inputs were fed into a model.)

For the regression task, the theoretical baseline MSE (calculated over the entire dataset if always predicting the mean value) is 0.0647. With any of our models, we were able to achieve a 0.0644 MSE.

For the classification task, we binned the data into three bins: positive (> 0.05), negative (< -0.05), and neutral. The bins had 33283, 100299, and 135927 examples, respectively; choosing the most probable bin (positive) yields a 50.44% accuracy. We were able to achieve a 50.38% accuracy.

These numbers are close enough to the baseline values that it's fairly obvious that it's predicting the average value (the slight variations are due to train/test splits). When manually observing the predicted values on the test dataset, we see that all of the values are closely clustered around the mean (in the regression case) or are almost all positive (in the classification case).

4 Conclusion

We were not able to achieve state of the art results for the sentiment analysis task. Our model was fairly simple and relied a lot on the BERT encoder and preprocessing TensorFlow pretrained models for the NLP-theoretic part of the project, but we probably implemented the encoder incorrectly. Our experimentation with the network architecture were not able to give us good insight into what our mistake was.

However, since the goal of this project was to learn how to use a deep learning library such as TensorFlow to apply to a NLP task, we were at least able to set up a runnable model and experiment with various model parameters, even if they didn't produce the results we intended to get.

Proposal for an independent study on the design of (mostly functional) languages

(a.k.a., a case for studying Scheme and *SICP*)

Jonathan Lam
to Prof. Sable

January 15, 2021

(I make a lot of references to the various resources in the section Readings to motivate LISP and *SICP*, so feel free to check that out first.)

1 Summary

This is a report intended to motivate an independent study on the design of a language, using Lisp as the object of consideration due to its metaprogramming capabilities.

2 Motivation (and responses to concerns)

My technical experience of the past roughly eight years comprises that of a: polyglot programmer; full-stack developer; (somewhat) dev-ops intern; computer center operator; programming mentor; founder of various high-school programming clubs; freelance web developer; computer hardware hobbyist; collector of old server hardware; independent driver developer; *nix power user; and, most recently student in deep learning and artificial intelligence. Through these roles I've become fairly convinced that the quality of one's code is paramount to how well a program performs; how performant, scalable, and maintainable the source tree is; and how easy it is for others to understand and contribute to a project. My concern is that, having viewed many peers' code via mentoring or group projects (or my coworkers' code when I was an intern), and having reflected on my own code from the past, that good coding quality standards are rarely taught nor enforced in many situations where a deadline must be met (particularly susceptible are academic projects and scientific computing). I wish for a more formal practice; to be able to think regularly in terms of better design principles and meaningfully-structured code.

The second issue is that I would like to pursue some topic in the space of operating systems, compilers, or language design: something in the infrastruc-

ture of the programming stack. I feel that my work has been saturated with an interest in the metaprocess of building programs; I always want to know how the libraries, the languages that comprise those libraries, and the systems that host those languages are built. In other words, I want to understand the series of abstractions, from low-level computing to the design of practical high-level libraries. In the end, I hope to perform research in this field for a graduate degree, but I am not sure what in this somewhat-large space on which to focus.

The intersection of these two topics is *Structure and Interpretation of Computer Programs* (*SICP*). The forward and prefaces to the book nicely declares its intent; here is an excerpt from the prologue by Alan Perlis:

> Unfortunately, as programs get large and complicated, as they almost always do, the adequacy, consistency, and correctness of the specifications themselves become open to doubt, so that complete formal arguments of correctness seldom accompany large programs. Since large programs grow from small ones, it is crucial that we develop an arsenal of standard program structures of whose correctness we have become sure – we call them idioms – and learn to combine them into larger structures using organizational techniques of proven value. These techniques are treated at length in this book, and understanding them is essential to participation in the Promethean enterprise called programming. More than anything else, the uncovering and mastery of powerful organizational techniques accelerates our ability to create large, significant programs.

That is, *SICP* is a book about abstraction as a building block of building practical software. Not only that, but the last two chapters specifically focus on designing a new LISP interpreter as an example of a large useful program built with the programming practices emphasized in the earlier parts of the book. The book uses LISP as the object of focus due to its powerful metaprogramming capabilities.

My goal is to turn this into an independent study over the summer so I can have a mentor to ease me through the teachings of this book, to have someone to discuss inquiries with, and to keep me focused. The ultimate goal is that this, along with the related exercises and readings, will help me explore into the field of language design principles so that I can smoothly transition into a research project by the fall semester.

I don't think I need to elaborate on why LISP such a fascinating language, both in its history and its structure, but I want to defend the argument that it (and *SICP*) can still be the language (and book) of choice today in an academic setting.

2.1 Modern irrelevance – Choice of book and language

The largest concern during our meeting last semester was that both LISP and *SICP* are old, and its study may not be the most fruitful. This issue seems

critical, given that many CS programs use languages that are more modern and relevant to practical programming, such as C++, Java, and Python; even the courses that have traditionally used LISP (and SICP) to demonstrate some fundamentals of programming (most notably CS61A at Berkeley and 6.001 at MIT) have switched to Python-based curricula. Similarly, *How to Design Programs* (*HtDP*) was written to address some of the pedagogical concerns about *SICP*. Similarly, it is concerning that many of the reviews of *SICP* come from over a decade ago.

However, I would like to dispel these general concerns. First of all, there are innumerable sources that praise *SICP* for its timeless and language-agnostic ability to change the way a programmer thinks about his code from the building blocks available in the language. With what little experience I have in Scheme, I truly feel this is the case – while barebones LISP cannot compete with high-performance languages (e.g., FORTRAN, C, C++), domain-specific languages (e.g., SQL), or domain-specific libraries (e.g., TensorFlow), there are many appraisals of LISP as teaching you how you can build abstractions in software using the building blocks those languages and libraries have to offer you.

To argue about the relevance of *SICP*, I hope that the fact that they are still being offered at major universities (e.g., 6.037 at MIT), albeit not as the introductory course, speaks for itself. (Similarly, Berkeley's 61A teaches Python in the spirit of *SICP*, and its official course name is still "Structure and Interpretations of Computer Programs.") I also believe that the reasons why these prestigious schools stopped teaching with LISP in their introductory courses (see "Programming by Poking: Why MIT stopped teaching SICP") says something about the changing nature of what it means to work in a CS-related job (i.e., now, it is about "poking" around software libraries rather than really doing the engineering from the most basic building blocks), but my goal is more similar to the original ideal: to be able to understand how to manipulate programming languages well enough from their most basic building blocks by means of abstraction. And that is exactly what *SICP* is about.

Despite its old age, LISP is also still innovating: just looking at Scheme, for example, we have the evolving standard. Currently, seven versions of the Scheme standard (currently at R6RS and R7RS) have been released. The language offers several features that have yet to become a part of any other mainstream language, such as first-class continuations or hygienic macros, and its macro system is still unrivaled among mainstream language (the only other languages I've heard it's comparable to are metaprogramming languages such as MetaLua). There are several Scheme variants that are actively maintained and optimized, such as Chez Scheme by Cisco; other LISP variants, such as Common Lisp and Clojure, have major industry support (and the latter is a JVM language, and thus is as up-to-date as the JVM platform is). Performance-wise, many LISP compilers and interpreters have gotten to the point where performance loss compared to high-performance languages like C is negligible, and most variants of LISP beat out other high-level interpreted languages such as Python (at least plain Python without its BLAS or CUDA integrations). (And for many projects, the computing power of most devices is more than sufficient

for any modern language.) Along with the general fact that you can easily build abstractions upon abstractions (according to "Lisp Is Too Powerful", it "allows people to invent their own little worlds") means that using an efficient declarative language like LISP is a way to prevent accruing technical debt.

I know that some of these claims are weak, but the original concern that *SICP* and *LISP* are out-of-date are also based on pure speculation. With a good deal of Internet searching I was unable to find an example of a modern language feature that LISP was incapable of doing, outside of the typical pitfalls of declarative languages (e.g., explicit memory management, array-based data structures such as hashtables, static typing, etc.), and was unable to find any major modern innovation in functional programming that LISP specifically cannot perform. However, to be complete, I have linked *The Journal of Functional Programming* as a possible resource for this independent study, which could be useful for contemporary research (but this may not be very relevant to LISP or the main task at hand).

As an aside, I mentioned (*HtDP*) was a textbook intended to address some of the academic shortcomings of *SICP*, but I believe that many of those changes are actually in conflict with what I intend to achieve. While I have not read "The Structure and Interpretation of the Computer Science Curriculum" (the paper describing the differences between the books) in detail, from a quick perusal it seems that *HtDP* was intended to be a more student-friendly introduction, with a more gradual learning curve and explicit teachings. From what I understand, it also does not go into all the fuss of developing an interpreter as an ultimate goal as *SICP* does, but rather focuses on the more basic design aspects. However, since *SICP* explicitly goes into the details of building an interpreter; since I have a basic knowledge in LISP; and since I am not averse to the challenging problems (rather, I believe that struggling through these problems slowly are key to learning a topic well), I believe that the classic *SICP* will be the better textbook to achieve my goal.

2.2 Devil's advocate: why *not* a more modern language?

Rather than defend LISP, I now present an opposing view: why *not* use a language like Python, JavaScript, or Scala, which are undoubtedly more relevant nowadays? What does LISP have to offer that the others don't?

This concern was brought up in our meeting last semester, and my response was that LISP is known as the "programmable programmer's language," and I mumbled something about its homoiconicity, and that you could easily extend the syntax of LISP. But your rebuttal is that you can't change the structure of LISP, any more than you can't change the structure of another language like C or Python. Plus, if we're talking about macros, C already has macros – what makes LISP any different or better?

I spent some time trying to clarify this via some Internet searches, many of which are in the Readings section below. My new answer to this is that the "programmable nature" of LISP is due to a combination of its homoiconic nature and its macros. The homoiconic nature means that the parsing of LISP

code is trivial due to its simple, recursive syntax that "can be summarized on a business card" (see "Lisp vs. Haskell") and very clearly corresponds to an AST. This is as opposed to many other languages, which have a plethora of inconsistent syntaxes to indicate different kinds of operations, which gives concision at the cost of terribly-inconvenient parsing. As a result of this, many languages that do have macros (recall that macros are functions which transforms source code; in addition to introducing syntactic sugar, they can also change control flow and thus introduce control (syntactic) primitives and otherwise) only have *lexical* macros, e.g., C's token-replacement preprocessor, which is nice for simple replacements but nothing more than that. LISP's *syntactic* macros, on the other hand, are well-integrated into the language and can perform arbitrary transformations (i.e., can run arbitrary logic on the input code to produce the output code, which is capable of infinitely many transformations and not simply text replacement), as well as provide features such as hygienic macros that are intractable (if not impossible) with lexical macros.

The fact that LISP has these syntactic macros makes it an ideal candidate for a language to build abstractions out of. No, we can't change the barebones syntax of LISP (i.e., parentheses, spaces, and tokens must follow the correct syntax), but we can define new syntactic primitives (e.g., control flow, text replacement, data-as-code, etc.) that allow us to extend the language. (If this is unclear, there are a number of resources in the Readings section which do a better job explaining this than I do here.) This undoubtedly was a key reason that the authors of *SICP* chose LISP, and I can see it as an excellent choice for experimenting with different language constructs and styles.

2.3 Relationship with functional languages?

From the video call earlier, it seemed like your impression of the book (and this independent study) was that it might be a study of functional languages – this is not true. I believe you mentioned that you might consider teaching a course on functional programming in the future at Cooper, which is an endeavor I admire as a good step towards creating better programmers – but that is not my goal here. While Lisp is (primarily) functional, that is neither its defining characteristic nor the reason why it is so powerful.

I gave a talk about Lisp last semester in an IEEExACM-organized educational session (rough transcription) because I love the language and hope to indoctrinate as many people as I can. In the original flyer. In the event poster, I called the event "An introduction to Scheme Lisp and Functional Programming," but when I was generating the presentation I was wondering if this was what I really wanted to convey. Functional programming indeed is something that I don't believe many Cooper students are adequately exposed to (the main culprit being firstly that C and Python being taught in terms in data structures (which are mostly imperatively-defined, see "Disadvantages of purely functional languages"), and secondly that scientific programming doesn't at all promote non-imperative-style coding practices), but it didn't seem to me that it was the selling feature of Lisp. After all, many modern languages are

multi-paradigm now (e.g., modern high-level dynamic programming languages like Python, Scala, JavaScript, Ruby, and even recent editions of traditionally-imperative languages like C++ and Java), so this is not at all impressive anymore, even if LISP was the pioneer in many of the topics in functional programming. Rather, I believe that the fact that Lisp's powerful syntax inherently follow a recursive, expression-based, and therefore declarative nature (since its inception in McCarthy's original paper) is a byproduct of its metaprogramming capabilities, and should be treated as such. As mentioned earlier, I have included *The Journal of Functional Programming* as a possible resource for this class, which should more than suffice innovations in functional programming, but this study is orthogonal to LISP (as it should be).

For a study of functional programming, being forced to work in a purely-functional language such as Haskell or Prolog (the latter of which is a Lisp) is probably a better choice. To address the first concern about modernness, there are some resources in the Readings section that illustrate that Haskell is still relevant and performant, despite being function and high-level.

2.4 Lack of knowledge of LISP for mentorship

You voiced your concern that your last experience with LISP was many years ago, and may not have enough experience with the language to be able to advise me if the independent study were to focus on it. However, my view as a polyglot programmer is that that fact should never really be a problem. Also, I firmly believe that having a second person to discuss ideas with, even if they aren't designated experts in the field, is incredibly useful. Perhaps discussions with someone who is the opposite of an expert – an inquisitive, but otherwise clueless on the matter – is the best way to learn, because then more naïve questions are posed and answered, and you simply double the brainpower. This is a fairly general and idealistic pedagogy, but I find discussions with both informed and uninformed people very useful when self-learning any material.

Of course, you (Prof. Sable) are not generally naïve when it comes to programming languages or the principles of data structures and algorithms (and I like to believe that I am also knowledgeable to some degree), so and this peripheral knowledge should accelerate the rate of learning.

2.5 CL vs. Scheme (vs. Others)

This is related to the concerns about the modernness and usefulness of Scheme, which is such a minimal language that it doesn't have many claims to fame outside of academia (and principally, outside of the realm of *SICP*). I remember you suggested that, instead of Scheme, we could use Common Lisp, which has many more libraries and is thus more "practical." However, I would like to argue this point on three grounds:

Firstly, that there is more than enough mental material to fill a semester (or two, or three) with purely academic exercises in Scheme, using only the exercises in *SICP* or additional exercises (e.g., from *The [Little/Intermediate/Seasoned]*

Schemer). Satisfactorily learning Common Lisp, a much more featured language with larger industry support, very well might be overwhelming for a semester

Secondly, that the ideas in *SICP* are largely language-agnostic (given that other general-purposes languages are less-capable of syntactic abstraction but more featured with their default syntax and libraries). Of course, modern vocational programming or scientific computing is highly language- and library-specific, and doesn't require the kind of analytical, problem-solving skillset that *SICP* offers. Perhaps that makes Scheme less appealing to students who simply want a quick job with their EE/CS degree, but again, that is not the goal for me here.

Thirdly, the fact that Scheme has not gained widespread appeal and is not useful for practical coding is probably largely true, but I have personally found it to be useful and somewhat widespread. In my casual programming experience, I have seen Scheme (mostly Guile Scheme due to its GNU integration) used in various useful projects, such as for xbindkeys (keyboard-to-macro mappings in Linux) and emacs lisp. Similarly, mainstream learning resources and coding platforms for Scheme is not at all limited: an excellent tool I have used in the past to learn new languages, exercism.io, includes a track for (Chez and Guile) Scheme. And for school and hobby projects (except perhaps for high-performance computing, such as deep learning and scientific computing that needs intimate control over memory), I've found Scheme to be plenty useful, e.g., for use in ECE469 Artificial Intelligence and MA352 Discrete Mathematics. I'm sure that I would have used Scheme in many other courses had I been familiar with it earlier.

2.6 Overlap with ECE466 (Compilers)

Knowing that *SICP* builds up towards writing a Lisp interpreter might make it seem to have significant overlap with a compilers course. However, a cursory look at the ECE466 course webpage indicates that the topics covered by this course are very different than the goals of *SICP*. While the compilers class shows how to build a "practical compiler," e.g., ASTs, parsing, control flow optimizations, etc., *SICP* is designed to illustrate how to build more complex applications by abstracting the building blocks of a fully-featured low-level language. In other words, it is not the end goal of Lisp to build a programming language compiler or interpreter, but it is a very illustrative example that shows the power of metaprogramming; this is reinforced by the view in "Short explanation of last two chapters in SICP." Furthermore, the methods with which *SICP* attempts to build a interpreter are fundamentally different from those of compilers, as the building blocks are very different: parsing and memory management are already implicit in Lisp, whereas any practical compiler design is lower-level and would need to take this into consideration. All in all, I hope that both ECE466 and this study will aid in selecting a research topic.

3 (Preliminary) (Tentative) Timeline

I believe that following the book should offer more than enough material for a course semester. As many testimonies of the book say, *SICP* is a tome that takes time to read and digest, and its exercises are not trivial. Even at prestigious universities such as Berkeley and MIT (at MIT it is taught by the original authors of the book), this is offered as a semester-long course (and the authors state in the forward that it is more material than what can be covered in a typical semester), and is used as the textbook for graduate-level courses on symbolic programming as well (see MIT's 6.945 in the Materials). We can follow a schedule similar to that posted on MIT or Berkeley's course webpages, and use the video lectures from MIT. Extra exercises may be pulled from *The [Little/Intermediate/Seasoned] Schemer*.

However, to satisfy the need to feel up-to-date, perhaps additional readings should be completed every week (in the tradition of Cooper's ECE472 Deep Learning course, perhaps a set of 2-5 weekly readings from *The Journal of Functional Programming*), or some of the other scholarly articles or books on Lisp listed below, may serve as additional reading.

I have already covered the first chapter of the book on my own, and have a working knowledge of the basics of LISP through school projects, so I believe we can dive straight into the second chapter. If we plan to cover the latter four chapters of the book for a roughly 13-week agenda for the summer semester, a possible weekly agenda for the independent study might look like:

1. §2.1-2.4: Symbolic data, data abstraction, data hierarchy and closures
2. §2.5: Systems with generic operations
3. Continuation of §2.5: Implement a symbolic algebra system (perhaps with macros and symbolic differentiation?)
4. §3.1-3.2: Local state and the evaluation model
5. §3.3-3.4: Mutable data and concurrency
6. §3.5: Streams (and perhaps a project to implement streams)
7. §4.1: The metacircular evaluator
8. §4.2-4.3: Lazy evaluation, nondeterministic computing
9. §4.4: Logic programming: (perhaps a project to implement a Prolog-like language)
10. §5.1-5.2: Register machines
11. §5.3: Allocation and garbage collection
12. §5.3: The explicit-control evaluator
13. §5.4: Compilation

4 (Preliminary) (Tentative) Deliverables and Assessment

Assignments from *SICP* and *The [Little/Intermediate/Seasoned] Schemer* could be used as weekly assignments, as well as reading reports. The final project may be a Lisp interpreter written in Lisp, following the trajectory of the book.

5 Materials

- *Structure and Interpretation of Computer Programs* (*SICP*)

  a.k.a. the "Wizard book," and the subject of discussion of most of this article

- Structure and Interpretation of Computer Programs (University of California, Berkeley CS61A)

  Python in the tradition of *SICP* (Associated materials: Composing Programs)

- Structure and Interpretation of Computer Programs (MIT 6.001)

  Original course taught by the authors of the book (stopped being taught in early 2000's)

- Structure and Interpretation of Computer Programs (MIT 6.037)

  Newer course taught by the authors of the book (2019)

- Adventures in Advanced Symbolic Programming (MIT 6.945)

  Graduate course on symbolic programming, uses *SICP* as its textbook

- Journal of Functional Programming

  If we want to stay up to date, can choose a few articles out of this every week. Unfortunately may need to pay to subscribe

- *On Lisp*

  A whole book on Lisp by Paul Graham (who is described in the article "How Lisp became God's Own Programming Language" below)

- *The Common Lisp Cookbook*

  The title is self-explanatory

- R6RS

  Latest widely-adopted Scheme standard, including standard libraries

- Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I

  Original paper by McCarthy on S-expressions, which formed the basis for Lisp; Lisp was implemented shortly after this paper using these principles

- *The [Little/Intermediate/Seasoned] Schemer*

  A series of problems, separated by difficulty, designed to illustrate various aspects of the Scheme language

- *Let over Lambda: 50 years of Lisp*

  A guidebook to Common Lisp, including more advanced features that you wouldn't normally find elsewhere; first six chapters are available online for free

- *How to Design Programs* (HtDP)

  A book mostly in the tradition of *SICP*, but specifically aimed at its pedagogical shortcomings (e.g., HtDP is intended to require less domain knowledge and be more explicit in its principles); The Structure and Interpretation of the Computer Science Curriculum addresses these pedagogical differences. Second edition came out last year (2018)

- *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp* (PAIP)

  Chapters include logic programming and other practical problems in Lisp (e.g., a general purpose solver (GPS), ELIZA, a computer algebra system (CAS) and other symbolic mathematics)

6 Readings to motivate LISP and *SICP*

I realize I skimmed quite a few Q&A posts, articles, books, etc. related to "Why Lisp?" and especially "Why Lisp macros (as opposed to macros in other languages or ordinary first-class functions)?" because macros are what really make Lisp so useful. Here is a brief summary of some articles.

- "Lisp: A language for stratified design"

  1987 research paper by the authors of *SICP* about the usefulness of Lisp as a way to show a complete view on programming techniques through abstraction

- "Scheme vs. Python"

  The title is misleading; this is written by the professor for 61A at Berkeley, which uses *SICP*, and it defends the use of the book and Scheme rather than Python. (Since this post was written, I believe 61A has switched to Python (most likely due to a reason similar to "Programming by Poking: Why MIT stopped teaching SICP"), but it still attempts to follow the tradition of *SICP*)

- "Is there an expiration date for well regarded, but old books on programming?"

No, most of the concepts in these books are fundamental (and therefore timeless), just as the basics of mathematics (e.g., calculus) hasn't changed for centuries; also many concepts are language-agnostic

- "Programming by Poking: Why MIT stopped teaching SICP"

  (The title is misleading: MIT still has courses that use and teach *SICP*, but the iconic 6.001 course that was the introduction to all EECS undergrads at MIT was abolished in favor of an introductory course in Python, 6.0001)

  Unfortunately, much of the real work programmers do nowadays is "programming by poking": messing around with pieces of code (high-level languages and libraries) to just get it to work, people don't have to build systems from the bottom up nowadays, so they switched the introductory course to python

  Personally, I find this reason very sad and reflects the state of most programmers nowadays. But this is contradictory to my motivation (see section above)

- "Notes on Structure and Interpretation of Computer Programs"

  Mentions a few notable topics that *SICP* teaches: homoiconicity, how to write an interpreter, register machines, etc.

- "Why is Haskell (GHC) so darn fast?"

  A discussion on the speed of modern functional languages, and how they achieve their impressive real-world speed (as opposed to the common mode of thinking that functional languages are typically more inefficient)

- "Disadvantages of purely functional languages"

  Written by a pronounced naysayer of Haskell; mostly describes the fact that some common data structures cannot be implemented in a functional way

- "Lisp vs. Haskell"

  Speaks to Lisp's metaprogramming abilities and simplicity ((almost) everything is built out of seven core functions)

- "What is so great about Lisp?"

  General discussion on why so many people admire Lisp. Mostly emphasizes on the simple syntax, which leads naturally to a metacircular interpreter, as well as bringing up a host of interesting readings and Greenspun's tenth rule

  Also mentions a few disadvantages, such as the relatively small number of libraries, difficulty to pick up, and being forced to use dynamic typing (whether this is a boon or poison depends on the application). However, none of these affect learning the fundamentals in an academic context

- "Lisp Is Too Powerful"

  The first sentence is illustrative: "I think one of the problems with Lisp is that it is too powerful. It has so much meta ability that it allows people to invent their own little worlds, and it takes a while to figure out each person's little world"

- "What makes Lisp a programmable programming language? Why does this matter? What are the advantages of being one?"

  Describes what it means when people describe Lisp as a "programmable programming language" – namely, that it can

  Also describes how Haskell doesn't achieve these goals, which shows that (even purely) functional programming does not amount to the metaprogramming capability that Lisp offers. Is probably more true for less-functional languages like Python

- "How Lisp became God's Own Programming Language"

  Perhaps my favorite articles on "Why Lisp" that feels complete in multiple ways. Goes into a history of Lisp (including its motivations and impacts). The last paragraph, about the revival of Lisp after the AI winter and its lingering impact on "modern" languages like Ruby and Python, is especially satisfying.

- "What makes Lisp macros so special?"

  Many good answers here. E.g., can implement list comprehension or infix notation in Lisp, write code replacements (which is more advanced than the C's token-expansion, write something like C#'s LINQ notation; a macro is a function that takes in some source code and outputs source code, but it can perform logic on the expansion as well, including processing the arguments to the macro), changing execution order, the fact that Lisp's homoiconic nature makes macros much more useful than other languages with macros ("in C, you would have to write a custom preprocessor [which would probably qualify as a sufficiently complicated C program]")

- "What are Lisp macros good for, anyway?"

  Illustrative example of a timing macro in Lisp, and comparison to purely functional approaches (i.e., without macros) in Java and Python. Illustrates how macros allow for a different evaluation pattern for its arguments, i.e., rewriting the evaluation path, which cannot be done simply with functions (i.e., we can create new control primitives just like builtin control flow operators)

- "What can you do with Lisp macros that you can't do with first-class functions?"

  Relevant to the previous resource. Useful comment on an answer: "To abstract and hopefully clarify a bit, Lisp macros allow you to control

whether (and when) arguments are evaluated; functions do not. With a function, the arguments are always evaluated when the function is invoked. With a macro, the arguments are passed to the macro as ordinary data. The macro code then decides itself if and when they should be evaluated. This is why you can't write reverse_function without macros: as a function, the argument would be evaluated (and cause an error) before the function body gets a chance to rewrite it."

This also goes over some of the basic ideas, e.g., that macros do expansion at compile-time whereas first-class functions create closures at runtime and have the extra overhead of a function call; this has the added benefit of greater efficiency (similar to macros in any other language)

- Paul Graham's essays (mostly on programming languages)

  Haven't read these yet, but a good number are on Lisp. Paul Graham is a co-founder of ycombinator and helped to revive Lisp after the AI winter with these essays (for more history, see "How Lisp became God's Own Programming Language")

- "Automata via Macros"

  Research paper aiming to bring light to the relevance and need of Lisp-like macros, since they are not widely used in more modern languages (e.g., Java and C). Also focuses on the need for tail-call optimization for macros to be useful and efficient.

  Also has a very succinct list of overview of uses of macros: "providing cosmetics; introducing binding constructs; implementing 'control operators', i.e., ones that alter the order of evaluation; defining data languages"

- "Why do people say Lisp has true macros versus C/C++ macros?"

  A question dedicated to the comparison of Lisp-style macros to those that exist in other languages. The main point is that since the language is homoiconic, you have the power of the entire Lisp language at your hands to transform the input code to the output code, whereas in C/C++ the language is much more complicated and only does token replacements. Similarly, the C preprocessor is very decoupled from the rest of the language and doesn't have any logical utilities (other than token replacement), whereas the Lisp macro system is very intimately tied into its language, as you can perform arbitrarily-complex Lisp code to perform the code transformation

- "Lisp Macro"

  Similar arguments as in the previous articles, with some personal testimonies. Includes a lot of Q&A about use cases (including when not to use macros)

- "Why aren't macros included in most modern programming languages?"

Yet another explanation of macros and how they are different in C and Lisp. While C macros are lexical and carry no syntactic meaning, Lisp macros are syntactic and can be treated as regular code. (E.g., one implication is the use of hygienic macros in Lisp, which safely do not shadow variables in the outside scope)

- "What makes macros possible in lisp dialects but not in other languages?"

  An answer to this question with some technical explanation

- "Short explanation of the last two chapters of SICP"

  Explains that the final two chapters are not a repeat of a compilers course, but show that an interpreter is just a computer program that takes data as input, and thus is a good example of data abstractions

- "SICP - Worth it?"

  Relevant quote: "It's a subtle book. As others say, you do things in it you probably won't ever do again, in a language you'll probably never use again... But it can change (and improve) the way you write and understand programs, in any language."

MA347 – HW1

Jonathan Lam

January 20, 2021

For these questions, we assume that $\mathbb{N}$ and $\mathbb{Z}$ are closed over addition and multiplication, and multiplication distributes over addition. and define $-\mathbb{N} \coloneqq \{-n : n \in \mathbb{N}\}$, just like we did in class.

1. Prove that there is no integer between 0 and 1.

   Lemma. *Let $a \in \mathbb{N}$, $b \in -\mathbb{N}$. Then $ab \in -\mathbb{N}$. (Or, more simply, if $a > 0$, $b < 0$, then $ab < 0$.)*

   Proof of lemma. Let $c = -b \in \mathbb{N}$. Then $ac \in \mathbb{N}$. Then:

   $$ab + ac = a(b + c) = a(b + (-b)) = a(0) = 0 \Rightarrow 0 - ab = ac \in \mathbb{N}$$

   Thus $ab < 0$. □

   Proof. Let $S = \{s \in \mathbb{Z} : 0 < s < 1\}$, and assume that S is nonempty (i.e., that there exists an integer between 0 and 1). Since S consists only of positive integers, $S \subseteq \mathbb{N}$. By WOP there exists a least element $n_0 \in S$.

   Now, let us examine $n_0^2 = n_0 n_0$. Since $\mathbb{N}$ is closed over multiplication, $n_0^2 \in \mathbb{N}$, and $n_0 < 1 \Rightarrow n_0 - 1 < 0$:

   $$n_0^2 - n_0 = n_0(n_0 - 1) < 0 = n_0 - n_0$$

   by the lemma above. Adding n_0 to both sides, we get $n_0^2 < n_0$. However, this contradicts the assertion that n_0 is a least element in $\mathbb{N}$, and thus S must be empty. Thus there exist no integers that lie between 0 and 1. □

2. (Page 5 no. 4) Prove

$$\prod_{k=1}^{n}\left(1+\frac{1}{k}\right)^{k}=\frac{(n+1)^{n}}{n!}$$

Proof. Let the hypothesis be called $A(n)$. The base case is $n=1$:

$$\left(1+\frac{1}{1}\right)^{1}=2=\frac{(1+1)^{1}}{1!}$$

and thus $A(1)$ is true. Inductive step:

$$\begin{aligned}\prod_{i=1}^{k+1}\left(1+\frac{1}{i}\right)^{i}&=\left[\prod_{i=1}^{k}\left(1+\frac{1}{i}\right)^{i}\right]\left(1+\frac{1}{k+1}\right)^{k+1}\\&=\left(\frac{(k+1)^{k}}{k!}\right)\left(\frac{k+2}{k+1}\right)^{k+1}\\&=\frac{(k+2)^{k+1}}{(k+1)k!}\\&=\frac{((k+1)+1)^{(k+1)}}{(k+1)!}\end{aligned}$$

and thus $A(k)$ is true implies that $A(k+1)$ is true. By induction first form, $A(n)$ is true $\forall n\in\mathbb{N}$. □

MA347 – HW2

Jonathan Lam

January 21, 2021

Page 9 #1 and Page 13 #1

1. Prove that there are infinitely many prime numbers.

   Proof. Let $\{p_i\} = 2, 3, \ldots, P$ be the set of primes up to and including P. Let $N = \prod_i p_i + 1$. By the fundamental theorem of arithmetic, any natural number $n > 2$ must be factorable into a product of primes, i.e., N must have a prime factor. It suffices to prove that any prime dividing N is greater than P; then, given any prime P, we can prove the existence of a larger prime using this method.

   Let d be a prime that divides N, and $N = qd$. Rearranging:

   $$N - \prod_i p_i = 1$$
   $$qd - \prod_i p_i = 1$$
   $$qd - \left[\prod_{i \neq j} p_i\right] p_j = 1$$

   From this we see that 1 is in the ideal generated by d and p_j, for any prime $p_j \leq P$, and thus d is relatively prime with all primes no greater than P. Thus any prime factor of N must be larger than P. □

2. Let $n, d \in \mathbb{N}$ and assume $1 < d < n$. Show that n can be written in the form

$$n = c_0 + c_1 d + \cdots + c_k d^k$$

with integers c_i such that $0 \leq c_i < d$, and that these integers c_i are uniquely determined.

Proof. To show existence, since we have $n, d \in \mathbb{N}$, we can use the Euclidean algorithm to find c_0, n_1 such that

$$n = c_0 + n_1 d \tag{1}$$

where $0 \leq c_0 < d$. Similarly, we can use the Euclidean algorithm on the quotient n_1:

$$n_1 = c_1 + n_2 d \tag{2}$$

with a similar constraint on c_1; we can keep iterating (2) using n_i, d and the Euclidean algorithm to find c_i, n_{i+1} until the quotient n_i becomes zero. (With every step, the quotient becomes smaller, since $qd \leq n$ and $d > 1$, so the quotient must reach zero after some finite number of iterations.) After substituting and simplifying the expression, n is expressed in the form shown in the hypothesis, in which each coefficient fits the constraint $0 \leq c_i < d$; i.e., this construction leads to an expression of the form:

$$n = c_0 + (c_1 + (c_2 + (\cdots + c_k d)d)d)d = c_0 + c_1 d + c_2 d^2 + \cdots + c_k d^k$$

To show uniqueness, we use the second form of induction. The base case is that c_0 is uniquely determined by the Euclidean algorithm in (1). For the inductive step, assume that $\{c_i\}_1^r$ are uniquely determined, and show that c_{r+1} is then determined. We know that c_i, d uniquely determines n_{i+1} by the Euclidean algorithm, so $\{n_{i+1}\}_1^r$ are also uniquely determined. Thus we have $n_{r+1}, d \in \mathbb{N}$ both uniquely determined, which uniquely determine c_{r+1}. Thus every $c_i \in \{c_i\}_1^k$ is uniquely determined. $\square$

MA347 – HW3

Jonathan Lam

January 27, 2021

(In these solutions, we take for granted that the rules of matrix multiplication (e.g., associativity), matrix inversion (e.g., if $T, U \in M_{n\times n}(\mathbb{R})$ are invertible, then $(TU)^{-1} = U^{-1}T^{-1}$), and integer addition (e.g., integers are closed over additive inverse and addition/subtraction) are known.)

1. Define $A \underset{s}{\sim} B \Leftrightarrow A = TBT^{-1}$ for some invertible $T \in M_{n\times n}(\mathbb{R})$, and $A, B \in M_{n\times n}(\mathbb{R})$. Prove that $\underset{s}{\sim}$ is an equivalence relation on $M_{n\times n}(\mathbb{R})$.

   Proof. **Reflexivity** Let $A \in M_{n\times n}(\mathbb{R})$. It is clear that $I_n \in M_{n\times n}(\mathbb{R})$, and $I_n^{-1} = I_n$. $A = IA = (IA)I = (IA)I^{-1} = IAI^{-1} \Rightarrow A \sim A$.

   Symmetry Let $A, B \in M_{n\times n}(\mathbb{R})$, and let $T \in M_{n\times n}(\mathbb{R})$ invertible. Then $A = TBT^{-1} \Rightarrow T^{-1}AT = T^{-1}TBT^{-1}T \Rightarrow IBI = B = T^{-1}AT$. Since T is invertible, then $U = T^{-1}$ is also invertible matrix in $M_{n\times n}(\mathbb{R})$, where $(T^{-1})^{-1} = T$. Thus $B = UBU^{-1}$, where $U \in M_{n\times n}(\mathbb{R})$ invertible; thus $B \sim A$.

   Transitivity Let $A, B, C \in M_{n\times n}(\mathbb{R})$, and let $T, U \in M_{n\times n}(\mathbb{R})$ invertible, such that $A = TBT^{-1}, B = UCU^{-1}$ (i.e., $A \sim B, B \sim C$). Then $A = T(UCU^{-1})T^{-1} = (TU)C(U^{-1}T^{-1})$. If we let $V = TU$, then we know that TU is in $M_{n\times n}(\mathbb{R})$ invertible and $(TU)^{-1} = U^{-1}T^{-1}$; thus $A = VCV^{-1} \Rightarrow A \sim C$.

   □

2. Define $x \sim y \Leftrightarrow x - y = n$ for some $n \in \mathbb{Z}$ and that $x, y \in \mathbb{R}$. Prove that $\sim$ is an equivalence relation on $\mathbb{R}$.

   Proof. **Reflexivity** Let $x \in \mathbb{R}$. Then $x - x = 0 \in \mathbb{Z} \Rightarrow x \sim x$.

   Symmetry Let $x \sim y$, where $x, y \in \mathbb{R}$. Then $x - y \in \mathbb{Z} \Rightarrow y - x = -(x - y) \in \mathbb{Z} \Rightarrow y \sim x$.

   Transitivity Let $x, y, z \in \mathbb{R}$, and $x \sim y$, $y \sim z$. Then $x - y = n_1 \in \mathbb{Z}$, $y - z = n_2 \in \mathbb{Z} \Rightarrow y = z + n_2 \Rightarrow x - (z + n_2) = n_1 \Rightarrow x - z = n_1 - n_2 \in \mathbb{Z} \Rightarrow x \sim z$.

   □

MA347 – HW4

Jonathan Lam

February 4, 2021

1. Prove that for $b, g, h, m, n \in \mathbb{Z}$, $b|g$ and $b|h \Rightarrow b|mg + nh$.

   Proof. $b|g \Rightarrow g = xb$ for some $x \in \mathbb{Z}$, and $b|h \Rightarrow h = yb$ for some $y \in \mathbb{Z}$. Thus $mg + nh = m(xb) + n(yb) = (mx)b + (ny)b = (mx + ny)b \Rightarrow b|mg + nh$. □

2. For p prime and $[x]_p \neq [0]_p$, prove that there is $[y]_p \in \mathbb{Z}/p\mathbb{Z}$ such that $[x]_p[y]_p = [1]_p$. (I.e., prove that there exists a multiplicative inverse in the quotient group formed by the modulus.)

   Proof. Let $a \in [x]_p$ (i.e., $a \equiv x \bmod p$). Since the product is well-defined (i.e., $[x]_p[y]_p = [xy]_p$), we need only prove that $\exists m$ such that $am \equiv 1 \pmod p$ (i.e., that $am \in [1]_p$).

   $a \notin [0]_p \Rightarrow p \nmid a$. Since the only factors of p are 1 and p, $\gcd(a, p) = 1$. Then 1 exists in the ideal generated by a and p, i.e., $\exists m, n \in \mathbb{Z}$ such that $ma + nb = 1 \Rightarrow ma \equiv 1 \pmod p \Rightarrow ma \in [1]_p$.

   Choose $[y]_p = [m]_p$, i.e., the modular equivalence class of which m is a member. Then $[x]_p[y]_p = [a]_p[m]_p = [am]_p = [1]_p$. □

MA347 – HW5

Jonathan Lam

February 4, 2021

1. Let G be a group and $a, b \in G$. Prove that $(aba^{-1})^n = ab^n a^{-1}$ for each $n \in \mathbb{N}$.

   Proof. Let $A(n)$ be the proposition that this equation is true when the exponent is n. $A(1)$ is clearly true, since $(aba^{-1})^1 = aba^{-1} = ab^1a^{-1}$.

   Assume that this is true for $A(n)$ for some $n \in \mathbb{N}$. Then $(aba^{-1})^{n+1} = (aba^{-1})^n(aba^{-1}) = (ab^n a^{-1})(aba^{-1}) = ab^n(a^{-1}a)ba^{-1} = ab^n eba = ab^n ba = ab^{n+1}a$, where e is the unit element of G. Thus $A(n)$ is true $\Rightarrow A(n+1)$ is true. By the first form of induction, $A(n)$ is true $\forall n \in \mathbb{N}$. □

2. Let $\mathbb{K} = \mathbb{Q}, \mathbb{R}$, or $\mathbb{C}$. For each $a \in \mathbb{K}^n$ and $A \in \mathrm{GL}(n, \mathbb{K})$. Define $f_{A,a} : \mathbb{K}^n \to \mathbb{K}^n$ by $f_{A,a}(x) = Ax + a$.

   (a) Prove that $f_{A,a}$ is a bijective map.

   (b) Let $\mathrm{Aff}(n, \mathbb{K}) = \{f_{A,a} : A \in \mathrm{GL}(n, \mathbb{K}) \text{ and } a \in \mathbb{K}^n\}$. Prove that $\mathrm{Aff}(n, \mathbb{K})$ is a group under composition operation.

   Proof. (a) Prove bijective by proving both one-to-one and onto.

   Injectivity Let $x, y \in \mathbb{K}^n$, and $f_{A,a}(x) = f_{A,a}(y)$. Then:

   $$\begin{aligned} f_{A,a}(x) = Ax + a \Rightarrow x &= A^{-1}(f_{A,a}(x) - a) \\ f_{A,a}(y) = Ax + a \Rightarrow y &= A^{-1}(f_{A,a}(y) - a) \\ &= A^{-1}(f_{A,a}(x) - a) \\ &= x \end{aligned}$$

   $f_{A,a}(x) = f_{A,a}(y) \Rightarrow x = y$, thus injective.

   Surjectivity Let $x \in \mathbb{K}^n$. Let $y = A^{-1}(z - a) \in \mathbb{K}^n$. Then

   $$\begin{aligned} f_{A,a}(y) &= A(A^{-1}(z-a)) + a \\ &= (AA^{-1})(z-a) + a \\ &= I_n(z-a) + a \\ &= (z-a) + a = z + (-a + a) = z + 0 = z \end{aligned}$$

   Thus $\forall x \in \mathbb{K}^n$ there exists $y \in \mathbb{K}^n$ such that $f_{A,a}(y) = x$, thus surjective.

(b) Show that $(\mathrm{Aff}(n,\mathbb{K}),\circ)$ is a group by showing that it follows the group axioms:

Associativity Composition of mappings of a set onto itself is always associative. In particular, let $f_1, f_2, f_3 \in \mathrm{Aff}(n,\mathbb{K})$. Let $x \in \mathbb{K}^n$. Then:

$$\begin{aligned}(f_1 \circ (f_2 \circ f_3))(x) &= f_1((f_2 \circ f_3)(x)) \\ &= f_1(f_2(f_3(x))) \\ &= (f_1 \circ f_2)(f_3(x)) \\ &= ((f_1 \circ f_2) \circ f_3)(x)\end{aligned}$$

Thus $f_1 \circ (f_2 \circ f_3) = (f_1 \circ f_2) \circ f_3$.

Identity element Let I_n denote the identity element of $\mathrm{GL}(n,\mathbb{K})$, and 0_n denote the zero element of $\mathbb{K}^n$. Let $e = f_{I_n,0_n}$. Then for any $f_{A,a} \in \mathrm{Aff}(n,\mathbb{K})$ and any $x \in \mathbb{K}^n$:

$$\begin{aligned}(e \circ f_{A,a})(x) &= e(Ax + a) \\ &= I_n(Ax + a) + 0_n \\ &= Ax + a + 0_n \\ &= Ax + a \\ &= f_{A,a}(x)\end{aligned}$$

and

$$\begin{aligned}(f_{A,a} \circ e)(x) &= f_{A,a}(I_n x + 0_n) \\ &= f_{A,a}(x + 0_n) \\ &= f_{A,a}(x)\end{aligned}$$

thus $e \circ f_{A,a} = f_{A,a} = f_{A,a} \circ e$ for any $f_{A,a} \in \mathrm{Aff}(n,\mathbb{K})$.

Inverse element For any $f_1 = f_{A,a} \in \mathrm{Aff}(n,\mathbb{K})$, there exists $f_2 = f_{A^{-1},-A^{-1}a} \in \mathrm{Aff}(n,\mathbb{K})$, and

$$\begin{aligned}(f_1 \circ f_2)(x) &= f_1(A^{-1}x - A^{-1}a) \\ &= A(A^{-1}x - A^{-1}a) + a \\ &= A(A^{-1}(x - a)) + a \\ &= (AA^{-1})(x - a) + a \\ &= I_n(x - a) + a \\ &= (x - a) + a \\ &= x + (-a + a) \\ &= x + 0_n = I_n x + 0_n = e(x)\end{aligned}$$

and

$$\begin{aligned}(f_2 \circ f_1)(x) &= f_2(Ax + a) \\ &= A^{-}1(Ax + a) - A^{-1}a \\ &= A^{-1}(Ax) + A^{-1}a - A^{-1}a \\ &= (A^{-1}A)x + A^{-1}(a - a) \\ &= I_n x + A^{-1}(0_n) = I_n x + 0_n = e(x)\end{aligned}$$

Thus $\forall f_1 \in \mathrm{Aff}(n, \mathbb{K})$, there exists $f_2 \in \mathrm{Aff}(n, \mathbb{K})$ such that $f_1 \circ f_2 = e = f_2 \circ f_1$.

□

MA347 – HW6

Jonathan Lam

February 8, 2021

1. (a) Let G be a group and $a \in G$. Suppose there is $n \in \mathbb{N}$ such that $x^n = e$. Prove that there is $m \in \mathbb{N}$ such that $x^{-1} = x^m$.

   If $n = 1$, then $x^1 = x = e$. Choose $m = 1$. Then $xx^1 = ee = e = ee = x^1 x \Rightarrow x^{-1} = x^1$.

   Otherwise, choose $m = n - 1 \in \mathbb{N}$. Then $xx^{n-1} = x^1 x^{n-1} = x^{1+n-1} = x^n = e$, and similarly $x^{n-1}x = x^{n-1}x^1 = x^{n-1+1} = x^n = e$, so $x^{-1} = x^{n-1}$.

   (b) Assume G is a finite group. Prove that given $x \in G$, there exists $n \in \mathbb{N}$ such that $x^n = e$.

   Assume not, i.e., given $x \in G$ for some finite group G, $\nexists n \in \mathbb{N}$ such that $x^n = e$. Then the set $S = \{x^1, x^2, x^3, \dots\} \subseteq G$ is infinite if all of the elements are distinct. If they are not all distinct, then $\exists$ distinct $i, j \in \mathbb{N}$ (w.l.o.g. assume $i < j$) such that $a^i = a^j \Rightarrow a^i = a^i a^{j-i} \Rightarrow a^{j-i} = e$. Then there exists $n = j - i \in \mathbb{N}$ such that $a^n = e$, but this contradicts the hypothesis.

   Thus all of the elements of S must be distinct, so it must be an infinite set, so G must be infinite-order $\Rightarrow$ contradiction of hypothesis. Thus there must $\exists n \in \mathbb{N}$ such that $x^n = e$.

2. Let G be a group such that $x^2 = e$ for all $x \in G$. Show that G is abelian.

   For all $x \in G$. By hypothesis, $xx = e = xx \Rightarrow x = x^{-1}$. Let $a, b \in G$. Then $ab = a^{-1}b^{-1} = (ba)^{-1}$ (this result was proved during lecture). Since $ba \in G$, $(ba)^{-1} = ba \Rightarrow ab = ba$.

MA347 – HW7

Jonathan Lam

February 15, 2021

1. Let G be a finite cyclic group of order n. Show that for each positive integer d dividing n, there exists a subgroup of order d.

   Proof. By definition, a cyclic group $\langle a \rangle$ is a group in which each element $x \in G$ can be written in the form $x = a^m$, for some $a \in G$ and $m \in \mathbb{N}$.

   We have already shown in a theorem during lecture that the order of a (finite) cyclic (sub)group $\langle a \rangle$ is the degree n of a (i.e., the smallest positive exponent of a), consisting of the (distinct) elements $G = \{e, a, a^2, \ldots, a^{n-1}\}$.

   Now suppose $d|n$, or $\exists q \in \mathbb{N}$ such that $n = dq$. Consider the subgroup generated by $a' = a^q = a^{n/d}$, i.e., $\langle a' \rangle \leq \langle a \rangle = G$. Then d is an exponent of a', since $a'^d = a^{(n/d)d} = a^n = e$. Furthermore, for some integer d' such that $0 < d' < d$, $a'^{d'} = a^m$, where $0 < qd' = m < n$, so $a^m \in G - \{e\}$ because the elements of the cyclic group are distinct. Thus d is the degree of a. By the theorem above, $|\langle a' \rangle| = d$. $\square$

2. Let G be a finite cyclic group of order n. Let a be a generator. Let r be an integer $\neq 0$, and relatively prime to n.

   (a) Show that a^r is also a generator of G.

   (b) Show that every generator of G can be written in this form.

   (c) Let p be a prime number, and G a cyclic group of order p. How many generators does G have?

Proof. (a) Since r, n are coprime nonzero integers, then there exists $x, y \in \mathbb{Z}$ such that $xr + (-y)n = 1 \Rightarrow xr = yn + 1$. Thus:

$$(a^r)^x = a^{rx} = a^{yn+1} = a^{yn}a = (a^n)^y a = e^y a = ea = a$$

Thus, for any $b = a^m \in G$ for some $m \in \mathbb{Z}$ such that $0 \leq m < n$, then b can be written as $(a^{rx})^m = (a^r)^{xm}$, so a^r is a generator for G.

(b) Assume that $b \in G$ cannot be written in the form a^r, where r is some nonzero integer coprime to n. Since $b \in G$, $b = a^{r'}$ for some $r' \in \mathbb{Z}$ not coprime to n, $0 \leq r' < n$.

Let $\gcd(r', n) = d > 1$. Then, $n/d, r'/d \in \mathbb{N}$, and

$$\left(a^{r'}\right)^{n/d} = a^{r'(n/d)} = a^{r'n/d} = a^{n(r'/d)} = (a^n)^{r'/d} = e^{r'/d} = e$$

Since $n, d \in \mathbb{N}$ and $d > 1$, then $n/d < n$. Thus b has a degree less than n, so the subgroup generated by b must have order less than n (by the theorem stated in the solution to (1)), so b cannot be a generator of G.

By contrapositive, if b is a generator of G, then it can be written in the form $b = a^r$, where r is a integer $\neq 0$ coprime to n.

(c) Each integer $m \in \mathbb{N}$ such that $0 < m < p$ is coprime to p. According to (a), each element of $\{a^m\}_1^{p-1} = G - \{e\}$ is a generator of G.

We can also show that e is not a generator of G (for any group with order > 1). For a finite cyclic group, the set of exponents form an ideal generated by the degree of a, i.e., the set of exponents is $J = \{xn : x \in \mathbb{Z}\}$ (theorem from lecture). Since $e = a^y$ for $y \in J$, and $d|y$, then e cannot be written in the form a^r for r, n coprime. By part (b), e is not a generator of G.

Thus G has $p - 1$ generators.

□

MA347 – HW8

Jonathan Lam

February 18, 2021

Let $n \in \mathbb{N}$ and $\mathbb{K} = \mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$. For $A \in \mathrm{GL}(n, \mathbb{K})$, define $f_{A,b} : \mathbb{K}^n \to \mathbb{K}^n$ by $f_{A,b}(x) = Ax + b$. Then $\mathrm{Aff}(n, \mathbb{K}) = \{f_{A,b} : A \in \mathrm{GL}(n, \mathbb{K}), b \in \mathbb{K}^n\}$ is a group under composition.

1. Define $\varphi : \mathrm{GL}(n, \mathbb{K}) \to \mathrm{Aff}(n, \mathbb{K})$ by $\varphi(A) = f_{A,0}$, where 0 is the zero element of $\mathbb{K}^n$. Prove that φ is a group homomorphism. Find $\operatorname{Ker} \varphi$.

   Proof of Homomorphism. We already know that $\mathrm{GL}(n, \mathbb{K})$ and $\mathrm{Aff}(n, \mathbb{K})$ are groups, so all that is left to show is that the mapping φ satisfies the equality:
   $$\varphi(AB) = \varphi(A) \circ \varphi(B) \quad \forall A, B \in \mathrm{GL}(n, \mathbb{K})$$
   Let $A, B \in \mathrm{GL}(n, \mathbb{K})$. Then, $\forall x \in \mathbb{K}^n$:
   $$\begin{aligned}
   (\varphi(AB))(x) &= ABx + 0 && \text{(def. } \varphi, f_{AB,0}) \\
   &= A(Bx) + 0 && \text{(associativity of group } \mathrm{GL}(n, \mathbb{K})) \\
   &= f_{A,0}(Bx) && \text{(def. } f_{A,0}, \varphi) \\
   &= (\varphi(A))(Bx + 0) && (0 \text{ is identity of } \mathbb{K}^n) \\
   &= (\varphi(A))((\varphi(B))(x)) && \text{(def. } f_{B,0}, \varphi) \\
   &= (\varphi(A) \circ \varphi(B))(x) && \text{(def. } \circ)
   \end{aligned}$$
   Since the result of $\varphi(AB)$ and $\varphi(A) \circ \varphi(B)$ match for all values of $x \in \mathbb{K}^n$, then $\varphi(AB) = \varphi(A) \circ \varphi(B)$. □

   The identity element of $\mathrm{Aff}(n, \mathbb{K}^n)$ is $e' = f_{I_n,0}$ (shown in HW5), so $\operatorname{Ker} \varphi = \varphi^{-1}(e') = \{I_n\}$ by inspection.

2. Define $\psi : \mathbb{K}^n \to \mathrm{Aff}(n, \mathbb{K})$ by $\psi(b) = f_{I_n, b}$, where I_n is the identity element of $\mathrm{GL}(n, \mathbb{K})$. Prove that ψ is a group homomorphism. Determine $\operatorname{Ker} \psi$.

   Proof of homomorphism. The method is the same as in the earlier problem, except that $\mathbb{K}^n$ is an additive group so we want the result (same homomorphism idea but denoted differently):

   $$\psi(a + b) = \psi(a) \circ \psi(b)$$

   Let $a, b \in \mathbb{K}^n$. Then $\forall x \in \mathbb{K}^n$:

   $$\begin{aligned}
   (\psi(a+b))(x) &= I_n x + (a + b) && \text{(def. } \psi, f_{I_n, a+b}) \\
   &= I_n x + (b + a) && \text{(commutativity of group } \mathbb{K}^n) \\
   &= (I_n x + b) + a && \text{(associativity of group } \mathbb{K}^n) \\
   &= ((\psi(b))(x)) + a && \text{(def. } f_{I_n, b}, \psi) \\
   &= I_n((\psi(b))(x)) + a && (I_n \text{ is identity of } K^n) \\
   &= (\psi(a))((\psi(b))(x)) && \text{(def. } f_{I_n, a}, \psi) \\
   &= (\psi(a) \circ \psi(b))(x) && \text{(def. } \circ)
   \end{aligned}$$

   Since the result of $\psi(a+b)$ and $\varphi(a) \circ \varphi(b)$ match for all values of $x \in \mathbb{K}^n$, then $\varphi(a + b) = \varphi(a) \circ \varphi(b)$. □

   Again, the identity element of the codomain is $e' = f_{I_n, 0}$, so $\operatorname{Ker} \psi = \psi^{-1}(e') = \{0\}$ by inspection.

MA347 – HW9

Jonathan Lam

February 25, 2021

1. Let G be a finite group. Assume $\text{Aut}(G) = \{I_G\}$. Prove that G is abelian and $x^2 = e\ \forall x \in G$.

   Proof. $\forall a \in G$, then the left conjugation map $c_a : G \to G$ defined by $c_a(x) = axa^{-1}$ lies in $\text{Inn}(G) \subseteq \text{Aut}(G) = \{I_G\} \Rightarrow c_a = I_G$. Thus, $\forall a, x \in G$:

   $$\begin{aligned} I_G(x) &= c_a(x) \\ x &= axa^{-1} \\ xa &= (axa^{-1})a \\ &= ax(a^{-1}a) \\ &= axe \\ &= ax \end{aligned}$$

   Thus G is abelian.

   The inverse map $f : G \to G$ defined by $f(x) = x^{-1}$ also lies in $\text{Aut}(G) = \{I_G\}$ (it is bijective since a group is closed over inverse and the inverse element is unique). Thus $f = I_G$, so $\forall x \in G\ f(x) = x^{-1} = x = I_G(x) \Rightarrow e = xx^{-1} = x^2$. □

2. Let G be a cyclic group and $f : G \to G'$ be a group homomorphism. Prove that $f(G)$ is a cyclic subgroup of G'.

   Proof. We proved in class that $f(G) \leq G'$ if f is a homomorphism, so we only need to show that $f(G)$ is cyclic.

   G is cyclic $\Rightarrow G = \langle a \rangle$ for some $a \in G$. Let $x' \in f(G)$. Then $x' = f(x)$ for some $x \in G$, where $x = a^n$, $n \in \mathbb{Z}$. We also proved in class that $\forall b \in G, n \in \mathbb{Z}\ f(b^n) = f(b)^n$ when given a group homomorphism f. Thus $x' = f(a^n) = f(a)^n$. Since every element of $f(G)$ can be written in the form $x' = f(a)^n$, $f(G)$ is a cyclic group generated by $f(a)$. □

MA347 – HW10

Jonathan Lam

February 27, 2021

Let $H \leq G$. Define $a \underset{H}{\sim} b$ iff. $b^{-1}a \in H$.

1. Prove that $\underset{H}{\sim}$ is an equivalence relation on G and $[a] = aH$, the left coset of H by a.

   Proof equivalence relation. Fix $H \leq G$.

   Symmetry Let $a, b \in G$ and $a \underset{H}{\sim} b$. Then

   $$\begin{aligned} a \underset{H}{\sim} b &\Rightarrow b^{-1}a \in H && \text{(def. } \underset{H}{\sim}\text{)} \\ &\Rightarrow (b^{-1}a)^{-1} = a^{-1}b \in H && \text{(GRP 3)} \\ &\Rightarrow b \underset{H}{\sim} a && \text{(def. } \underset{H}{\sim}\text{)} \end{aligned}$$

   Reflexivity Let $a \in G$. Then $a^{-1}a = e \in H$ (GRP 1).

   Transitivity Let $a, b, c \in G$, and $a \underset{H}{\sim} b, b \underset{H}{\sim} c$. Then:

   $$\begin{aligned} a \underset{H}{\sim} b &\Rightarrow b^{-1}a \in H \\ b \underset{H}{\sim} c &\Rightarrow c^{-1}b \in H \\ &\Rightarrow (c^{-1}b)(b^{-1}a) \in H && \text{(group closed over multiplication)} \\ &\Rightarrow c^{-1}(bb^{-1})a \in H && \text{(group operation is associative)} \\ &\Rightarrow c^{-1}ea = c^{-1}a \in H \\ &\Rightarrow a \underset{H}{\sim} c \end{aligned}$$

   The relation $\underset{H}{\sim}$ is symmetric, reflexive, transitive $\therefore$ equivalence relation. $\square$

Proof equivalence classes. We wish to show that

$$[a] = \{x \in G : a \underset{H}{\sim} x \Leftrightarrow x^{-1}a \in H\} = \{ah : h \in H\} = aH$$

Claim: $aH \subseteq [a]$. Proof: Let $x \in aH \Rightarrow x = ah$ for some $h \in H$. Then

$$\begin{aligned} a(a^{-1}h) &= (aa^{-1})h \\ &= eh = h \in H \\ &\Rightarrow x \underset{H}{\sim} a \Rightarrow a \underset{H}{\sim} x \Rightarrow x \in [a] \end{aligned}$$

Claim: $[a] \subseteq aH$. Proof: Let $x \in [a]$. Then

$$\begin{aligned} x \in [a] &\Rightarrow x \underset{H}{\sim} a \\ &\Rightarrow a \underset{H}{\sim} x && (\underset{H}{\sim} \text{ symmetric}) \\ &\Rightarrow a^{-1}x = h \in H \\ &\Rightarrow a(a^{-1}x) = ah,\ h \in H \\ &\Rightarrow x = ah,\ h \in H \\ &\Rightarrow x \in aH \end{aligned}$$

Thus we have $aH \subseteq [a] \subseteq aH \Rightarrow aH = [a]$. □

2. Let ${}^{S}/_{\sim}$ be the set of equivalence classes w.r.t. $\underset{H}{\sim}$. Find ${}^{S}/_{\sim}$.

   We found the set of equivalence classes of S w.r.t. $\underset{H}{\sim}$ in the previous question. Namely, this is the set of left cosets of H in G, i.e., ${}^{S}/_{\sim} = {}^{G}/_{H}$.

MA347 – HW11

Jonathan Lam

March 1, 2021

1. Let $\varphi_1 : \mathrm{Aff}(n, \mathbb{K}) \to \mathrm{GL}(n, \mathbb{K})$ be defined by $\varphi_1(f_{A,a}) = A$. Prove that φ_1 is a group homomorphism and find $\operatorname{Ker} \varphi_1$.

   Proof φ_1 homo. Let $f_1 = f_{A_1,a_1}, f_2 = f_{A_2,a_2} \in \mathrm{Aff}(n, \mathbb{K})$. Then, $\forall x \in \mathbb{K}^n$:

   $$\begin{aligned}
   (f_1 \circ f_2)(x) &= f_1(f_2(x)) \\
   &= f_1(A_2x + a_2) \\
   &= A_1(A_2x + a_2) + a_1 \\
   &= (A_1(A_2x) + A_1a_2) + a_1 && (L_{A_1} : \mathbb{K}^n \to \mathbb{K}^n \text{ is homo.}) \\
   &= A_1(A_2x) + (A_1a_2 + a_1) && (\text{associativity of } \mathbb{K}^n) \\
   &= (A_1A_2)x + (A_1a_2 + a_1) && (\text{associativity of } \mathrm{GL}(N, \mathbb{K})) \\
   &= f_{A_1A_2, A_1a_2 + a_1}
   \end{aligned}$$

   Thus $\varphi_1(f_1 \circ f_2) = \varphi_1(f_{A_1A_2, A_1a_2+a_1}) = A_1A_2 = \varphi_1(f_1)\varphi_1(f_2)$.

   $\therefore \varphi_1$ is a group homomorphism. □

   Finding the kernel of φ_1:

   $$\begin{aligned}
   f_{A,a} \in \operatorname{Ker} \varphi_1 &\Rightarrow \varphi_1(f_{A,a}) = e_{\mathrm{GL}(n,\mathbb{K})} \\
   &\Rightarrow A = I_n \\
   &\Rightarrow \operatorname{Ker} \varphi_1 = \{f_{I_n,a} : a \in \mathbb{K}^n\}
   \end{aligned}$$

MA347 – HW12

Jonathan Lam

March 9, 2021

1. Let G be a group. $a \in G$, $c_a : G \to G$ defined by $c_a(x) = axa^{-1}$. Then $c_a \in \mathrm{Aut}(G)$. $\mathrm{Inn}(G) = \{e_a : a \in G\}$ is a subgroup of G and $\varphi : G \to \mathrm{Aut}(G)$ defined by $\varphi(a) = c_a$ is a homo. Prove that:

   (a) $Z(G) = \operatorname{Ker}\varphi$ where $Z(G)$ is the center of φ.

   Proof.

   $$\begin{aligned}\operatorname{Ker}\varphi &= \{a \in G : \varphi(a) = e'\} \\ &= \{a \in G : c_a = I_G\} \\ &= \{a \in G : c_a(x) = I_G(x)\ \forall x \in G\} \\ &= \{a \in G : axa^{-1} = x\ \forall x \in G\} \\ &= \{a \in G : ax = xa\ \forall x \in G\} \\ &= Z(G)\end{aligned}$$

   □

   (b) $\mathrm{Inn}(G) \triangleleft \mathrm{Aut}(G)$. ($\mathrm{Out}(G) \coloneqq {}^{\mathrm{Aut}(G)}/_{\mathrm{Inn}(G)}$ is called the outer automorphism group.)

   Proof. Let $K = \mathrm{Inn}(G)$, $H = \mathrm{Aut}(G)$. We show the result by **NOR1**, i.e., that $xKx = K\ \forall x \in H$.

   Let $h = c_a \in \operatorname{Inn} G$. $x \circ h \circ x \in xHx^{-1}$. Then, $\forall y \in G$:

   $$\begin{aligned}(x \circ h \circ x^{-1})(y) &= (x \circ h)(x^{-1}(y)) \\ &= x(ax^{-1}(y)a^{-1}) \\ &= x(a)x(x^{-1}(y))x(a^{-1}) && (x \text{ is homo.}) \\ &= x(a)I_G(y)x(a^{-1}) && (\text{def. } x^{-1}) \\ &= (x(a))y(x(a))^{-1} && (x \text{ is homo.}) \\ &= c_{x(a)}(y)\end{aligned}$$

   thus $x \circ h \circ x \in H \Rightarrow xHx^{-1} \subseteq H$. Since $x^{-1} \in G$, $x^{-1}Hx \subseteq H \Rightarrow H \subseteq xHx^{-1}$.
   $\therefore xHx^{-1} = H$. □

2. Let $H \leq G$. Define $N_H \coloneqq N(H) = \{x \in G : xHx^{-1} = H\}$. Prove that:

   (a) xHx^{-1} is a subgroup of G.

   Proof. Let $a, b \in xHx^{-1} \subseteq G$. Then $a = xh_1x^{-1}$, $b = xh_2x^{-1}$ for some $h_1, h_2 \in H$, and

$$\begin{aligned} ab^{-1} &= (xh_1x^{-1})(xh_2x^{-1})^{-1} \\ &= (xh_1x^{-1})(xh_2^{-1}x^{-1}) \\ &= xh_1(x^{-1}x)h_2^{-1}x^{-1} \\ &= xh_1h_2^{-1}x^{-1} \end{aligned}$$

   Since H is a group and thus closed under the group operation and its inverse, $h_1h_2^{-1} = h_3 \in H$, and $ab^{-1} = xh_3x^{-1} \in xHx^{-1}$.

   $\therefore xHx^{-1} \leq G$. □

   (b) $N_H = N(H)$ is a subgroup of G and H is a normal subgroup of $N(H) = N_H$.

   Proof. ($N_H \leq G$) Let $a, b \in N_H \subseteq G$. Clearly $b^{-1} \in N_H$ as well since $bHb^{-1} = H \Leftrightarrow H = b^{-1}Hb$. Then $aH = Ha$ and $b^{-1}H = Hb^{-1}$, and

$$\begin{aligned} (ab^{-1})H &= (aH)(b^{-1}H) && \text{(def. product of normal subgroups)} \\ &= (Ha)(Hb^{-1}) && (a, b^{-1} \in N_H) \\ &= H(ab^{-1}) && \text{(def. product of normal subgroups)} \end{aligned}$$

   Thus $(ab^{-1})H(ab^{-1})^{-1} = H \Rightarrow ab^{-1} \in N_H$.

   $\therefore N_H \leq G$. □

   Proof. ($H \triangleleft N_H$) H is a group by hypothesis. If $h \in H \subseteq G$, then $hHh^{-1} = (hH)h^{-1} = Hh^{-1} = H \Rightarrow h \in N_H \Rightarrow H \subseteq N_H$. Thus $H \leq N_H$.

   H is normal in N_H by definition of N_H, since $\forall x \in N_H \subseteq G$, $xHx^{-1} = H$ (**NOR1**). □

   (c) $H \triangleleft G \Leftrightarrow N_H = G$.

   Proof. By **NOR1**, we have $N_H \triangleleft G \Leftrightarrow xHx^{-1} = H \ \forall x \in G$. This proves the claim both ways.

   ($\Rightarrow$) Let $H \triangleleft G$. Then we have $xHx^{-1} = x \ \forall x \in G$. Thus the condition for N_H is satisfied $\forall x \in G$, so $N_H = G$.

   ($\Leftarrow$) Let $N_H = G$. Then $xHx^{-1} = H \ \forall x \in N_H = G \Rightarrow H \triangleleft G$. □

MA347 – HW13

Jonathan Lam

March 11, 2021

1. N and M are normal subgroups of a group G. Prove that NM is a normal subgroup of G.

   Proof. Let $a, b \in NM$. Then $a = m_1 n_1$, $b = m_2 n_2$ for some $m_1, m_2 \in M$, $n_1, n_2 \in N$. Thus

   $$\begin{aligned} ab^{-1} &= (n_1 m_1)(n_2 m_2)^{-1} \\ &= n_1 m_1 m_2^{-1} n_2^{-1} \\ &= n_1 m_3 n_2^{-1} \\ &= n_1 (m_3 n_2^{-1}) \end{aligned}$$

   where $m_3 = m_1 m_2^{-1} \in M$. Since $M \triangleleft G$, then $m_3 n_2^{-1} \in M n_2^{-1} = n_2^{-1} M \Rightarrow \exists m_4 \in M$ such that $m_3 n_2^{-1} = n_2^{-1} m_4 \in M$. Substituting:

   $$\begin{aligned} ab^{-1} &= n_1 (m_3 n_2^{-1}) \\ &= n_1 (n_2^{-1} m_4) \\ &= (n_1 n_2^{-1}) m_4 \\ &= n_3 m_4 \in NM \end{aligned}$$

   where $n_3 = n_1 n_2^{-1} \in N$. $\forall a, b \in NM \ ab^{-1} \in NM \Rightarrow NM \leq G$.

   To show $NM \triangleleft G$, we show that $x(NH) = (NH)x \ \forall x \in G$ (**NOR1**).

   Suppose $xn_1 m_1 \in x(NH)$, where $x \in G$, $n_1 \in N$, $m_1 \in M$. Using the property that $N \triangleleft G$. $xn_1 \in xN = Nx \Rightarrow \exists n_2 \in N$ such that $xn_1 = n_2 x \in Nx$. Similarly, $\exists m_2 \in M$ such that $xm_1 = m_2 x \in M_x$. Thus:

   $$\begin{aligned} & x(n_1 m_1) \in x(NM) \\ &= (xn_1) m_1 \\ &= (n_2 x) m_1 \\ &= n_2 (x m_1) \\ &= n_2 (m_2 x) \\ &= (n_2 m_2) x \in (NM) x \end{aligned}$$

   Thus $x(NM) \subseteq (NM)x \ \forall x \in G$. Similarly, we can show that $(NM)x \subseteq x(NH) \Rightarrow (NM)x = x(NM) \ \forall x \in G$. $\quad \therefore NM \triangleleft G$ $\square$

2. Let G be a group and let S be the set of subgroups of G. Define, for H and K in S, $H \underset{c}{\sim} K$ iff. $\exists x \in G$ s.t. $xHx^{-1} = K$. (We say that H is conjugate to K if $H \underset{c}{\sim} K$.)

   (a) Prove that $\underset{c}{\sim}$ is an equivalence relation in S.

   Proof. **Reflexivity** Let $H \in S$. Then $\exists x = e \in G$ s.t.

$$\begin{aligned} xHx^{-1} &= eHe^{-1} \\ &= (eH)e \\ &= He && (e \in H,\ hH = H\ \forall h \in H) \\ &= H && (e \in H,\ Hh = H\ \forall h \in H) \end{aligned}$$

   $\therefore H \underset{c}{\sim} H$.

   Symmetry Let $H, K \in S$, $H \underset{c}{\sim} K$. Then $\exists x \in G$ s.t. $xHx^{-1} = K$. Then it is clear that $H = x^{-1}Kx = yKy^{-1}$. Thus $\exists y = x^{-1} \in G$ s.t. $yKy^{-1} = H \Rightarrow K \underset{c}{\sim} H$.

   Transitivity Let $H, K, L \in S$, and $H \underset{c}{\sim} K$, $K \underset{c}{\sim} L$. Then $\exists x, y \in G$ s.t. $xHx^{-1} = K$ and $yKy^{-1} = L$. Then:

$$\begin{aligned} L &= yKy^{-1} \\ &= y(xHx^{-1})y^{-1} \\ &= (yx)H(x^{-1}y^{-1}) && \text{(See below comment)} \\ &= (yx)H(yx)^{-1} \end{aligned}$$

   (Comment: It is intuitive that $y(xHx^{-1})y^{-1} = (yx)H(x^{-1}y^{-1})$, but to explicitly show that this is not simply an abuse of notation: this holds true because each element is of the form $yxhy^{-1}x^{-1}$ where $h \in H$, which belongs to both sets.)

   Thus $\exists z = yx \in G$ s.t. $zHz^{-1} = L \Rightarrow H \underset{c}{\sim} L$.

   $\underset{c}{\sim}$ is reflexive, symmetry, and transitive $\therefore$ equivalence relation. $\square$

   (b) Find the equivalence class $[H]_c$ of $H \in S$.

   By definition, $[H]_c = \{K \in S : \exists x \in G \text{ s.t. } xHx^{-1} = K\}$. Another way to describe each $K \in [H]_c$ is that $K = c_x(H)$, i.e., the image of H under conjugation by x (for some $x \in G$). Since c_x is a bijective homomorphism, then $H \underset{c}{\sim} K$ implies that $H \simeq K$, so the equivalence class includes all subgroups of G isomorphic to H under the conjugation map.

MA347 – HW14

Jonathan Lam

March 25, 2021

Let $\alpha = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 13 & 2 & 15 & 14 & 10 & 6 & 12 & 3 & 4 & 1 & 7 & 9 & 5 & 11 & 8 \end{pmatrix}$.

1. Write α as a product of disjoint cycles and find $|\alpha|$ (the period of α).

$$\alpha = (1 \quad 13 \quad 5 \quad 10)(3 \quad 15 \quad 8)(4 \quad 14 \quad 11 \quad 7 \quad 12 \quad 9)(2)(6)$$
$$|\alpha| = \operatorname{lcm}\{4, 3, 6, 1, 1\} = 12$$

(The 1-cycles are decorative as they are equal to ε, the identity map.)

2. Determine the sign of α.

Preliminary material: Using the notation from Lang, let $\epsilon : S_n \to \{1, -1\}$ denote the map from permutations of J_n to their sign. (-1 denotes an odd permutation, and 1 denotes an even permutation.) As noted in Thm. 6.4 of Lang, this is a group homomorphism.

Claim: an r-cycle is even iff. r is odd. I.e., $\epsilon(\sigma_r) = (-1)^{r+1}$, where σ_r is a r-cycle (problem II.§6.5 in Lang).

Proof. (Using induction) Let $\sigma = \begin{pmatrix} i_1 & \dots & i_r \end{pmatrix}$ be an r-cycle. Let $A(n)$ be the assertion that $\epsilon\left(\begin{pmatrix} i_1 & \dots & i_n \end{pmatrix}\right) = (-1)^{n+1}$, for $2 < n \leq r$.

Base case ($n = 2$): A transposition (2-cycle) is odd. Thus $A(2)$ is true.

Inductive step ($n > 2$, assume $A(n-1)$ is true): We have:

$$\begin{pmatrix} i_1 & \cdots & i_n \end{pmatrix} = \begin{pmatrix} i_1 & i_n \end{pmatrix} \begin{pmatrix} i_1 & \cdots & i_{n-1} \end{pmatrix}$$

Denote these as:

$$\sigma_n = \tau\sigma_{n-1}$$

We see this is true because σ_n and σ_{n-1} are almost the same, except that σ_{n-1} fixes n and $\sigma_{n-1}(i_{n-1}) = i_1$, while σ_n does maps $\sigma_n(i_{n-1}) = i_n$ and $\sigma_n(i_n) = i_1$. By composing τ with σ_{n-1}, we now have $(\tau\sigma_{n-1})(i_{n-1}) = \tau(i_1) = i_n$ and $(\tau\sigma_{n-1})(i_n) = \tau(i_n) = i_1$ while everything else is left unchanged, so $\tau\sigma_{n-1} = \sigma_n$.

Then by inductive hypothesis, $\epsilon(\sigma_{n-1}) = (-1)^(n-1+1) = (-1)^n$, and:

$$\begin{aligned}
\epsilon(\sigma_n) &= \epsilon(\tau\sigma_{n-1}) \\
&= \epsilon(\tau)\epsilon(\sigma_{n-1}) && (\epsilon \text{ is homo.}) \\
&= (-1)\epsilon(\sigma_{n-1}) && (\text{transposition is odd}) \\
&= (-1)(-1)^n && (\text{by inductive hyp.}) \\
&= (-1)^{n+1} \\
&\therefore A(n-1) \Rightarrow A(n)
\end{aligned}$$

By first form of induction, $A(n)$ is true for all $n \in \mathbb{N}$ for $2 \leq n \leq r$. In particular having $A(r)$ be true proves the claim. □

To find the sign of α, use the fact that ϵ is a homomorphism to decompose the sign into the product of the signs of the component cycles, and then use the above result to calculate the sign of each of the component r-cycles.

$$\begin{aligned}
\epsilon(\alpha) &= \epsilon\left(\begin{pmatrix} 1 & 13 & 5 & 10 \end{pmatrix}\begin{pmatrix} 3 & 15 & 8 \end{pmatrix}\begin{pmatrix} 4 & 14 & 11 & 7 & 12 & 9 \end{pmatrix}\right) \\
&= \epsilon\left(\begin{pmatrix} 1 & 13 & 5 & 10 \end{pmatrix}\right)\epsilon\left(\begin{pmatrix} 3 & 15 & 8 \end{pmatrix}\right)\epsilon\left(\begin{pmatrix} 4 & 14 & 11 & 7 & 12 & 9 \end{pmatrix}\right) \\
&= (-1)(1)(-1) = 1 \\
&\therefore \alpha \text{ is even}
\end{aligned}$$

MA347 – HW15

Jonathan Lam

March 28, 2021

(Problems II.§8.2, II.§8.3a from Lang)

1. Let $\pi : G \to \mathrm{Perm}(S)$ be a homo. where G is a group and S is a set. Prove that $\mathrm{Ker}\,\pi = \bigcap_{s\in S} G_s$ where G_s is an isotropy group of $s \in S$.

   Proof. Let $K = \mathrm{Ker}\,\pi, H = \bigcap_{s\in S} G_s$ for convenience.
   ($K \subseteq H$)

   $$\begin{aligned} K &= \{x \in G : \pi(x) = I_S\} \\ &= \{x \in G : xs = s\ \forall s \in S\} \\ &= \{x \in G : x \in G_s\ \forall s \in S\} \\ &\Rightarrow K \subseteq G_s\ \forall s \in S \\ &\Rightarrow K \subseteq H \end{aligned}$$

   ($H \subseteq K$) Assume not. Then $\exists x \in G$ such that $x \in H$ and $x \notin K$. Then

   $$\begin{aligned} x \notin K &\Rightarrow x \notin \{x \in G : \pi(x) = I_S\} = \{x \in G : xs = s\ \forall s \in S\} \\ &\Rightarrow \exists s \in S \text{ such that } xs \neq s \\ &\Rightarrow x \notin G_s \\ &\Rightarrow x \notin H \\ &\Rightarrow \text{contradiction} \Rightarrow H \subseteq K \end{aligned}$$

   $K \subseteq H \subseteq K \Rightarrow K = H$. □

2. Let G be a group of order p^n where p is prime and $n \in \mathbb{N}$. Prove that $Z(G)$ is nontrivial, i.e., that $|Z(G)| > 1$.

   By the class formula (Lang prop. 8.5), we have:

$$|G| = |Z(G)| + \sum_{i=1}^{m} (G : G_{y_i})$$

   where $\{y_i\}_{i=1}^{m}$ represent the conjugacy classes which contain more than one element, and $(G : G_{y_i}) > 1$ for $i = \{1, \ldots, m\}$.

   Since $|G| = (G : G_{y_i})|G_{y_i}|$, then $(G : G_{y_i})$ divides $|G| = p^n$, so $(G : G_{y_i}) = p^{r_i}$, where $1 \leq r_i < n$ for all $i = \{1, \ldots, m\}$.

   Using to the property that divisibility distributes over sums, we have:

$$\begin{aligned} &p \mid |G|,\ p \mid (G : G_{y_i})\ \forall i = \{1, \ldots, m\} \\ &\Rightarrow p \mid \left(|G| - \sum_{i=1}^{m} (G : G_{y_i}) = |Z(G)| \right) \end{aligned}$$

   Thus $|Z(G)|$ is a (non-zero) multiple of p.

MA347 – HW16

Jonathan Lam

March 28, 2021

1. Let $f : G \to G'$ be a group isomorphism. Let $g \in G$ and $k \in \mathbb{N}$ and $A = \{a \in G : a^k = g\}$ and $B = \{b \in G' : b^k = f(g)\}$. Prove that A and B have the same number of elements.

   $|A| = |B| \Leftrightarrow$ there exists a bijection $h : A \to B$. In particular, we show that the restriction of f to A meets this condition, i.e., that $h = f|_A : A \to G'$ such that h is defined by $h(x) = f(x)$.

   (All that remains to show is that $h(A) = B \subseteq G'$. Since f is injective, h is also injective; and any map is surjective onto its image.)

   Let $a \in A$. Then $a^k = g \Rightarrow f(a^k) = f(a)^k = g$ (since f is a homo.) $\Rightarrow f(a) = h(a) \in G$. Thus $h(A) \subseteq B$.

   Let $b \in B$. Since $f : G \to G'$ is surjective, $\exists a = f^{-1}(b) \in G$. f is an iso., so f^{-1} is also a homo. Then:

   $$g = f^{-1}(f(g)) = f^{-1}(b^k) = f^{-1}(b)^k = a^k$$
   $$\Rightarrow f^{-1}(b) \in A$$

   Thus $f^{-1}(B) \subseteq A \Rightarrow B \subseteq f(A) = h(A)$.

   $h(A) \subseteq B \subseteq h(A) \Rightarrow h(A) = B$. By the preceding logic, h is a bijection from A onto B, so $|A| = |B|$.

2. (Lang II.§8.4) Let G be a finite group acting on a finite set S.

   (a) For each $s \in G$, prove that $Gs = Gt$ if $t \in Gs$ and

$$\sum_{t \in Gs} \frac{1}{|Gt|} = 1$$

   Proof. ($Gs = Gt$) Let $s, t \in S$, and $t \in Gs$, and let π be the group homomorphism mapping $G \to \mathrm{Perm}(S)$. Then $t = (\pi(x))(s)$ for some $x \in G$. $\pi(x) \in \mathrm{Perm}(S)$ implies that there exists an inverse permutation $\in \mathrm{Perm}(S)$ such that $s = ((\pi(x))^{-1})(t) \in Gt$.
   Thus $t \in Gs \Rightarrow s \in Gt$ so $Gs \subseteq Gt$, and $s \in Gt \Rightarrow t \in Gs$ by the same argument so $Gt \subseteq Gs$. Thus $Gs = Gt$. □

   Then we have $\forall t \in Gs$, $|Gt| = |Gs|$, so

$$\sum_{t \in Gs} \frac{1}{|Gt|} = |Gs| \frac{1}{|Gs|} = 1$$

   (b) Show that the number of orbits of G is S is equal to

$$\sum_{s \in S} \frac{1}{|Gs|}$$

   We know (by the previous result that $t \in Gs \Rightarrow Gs = Gt$) that the orbits form a partition over S. Let $\{y_i\}_{i=1}^m$ represent the distinct orbits of S. Then we can rewrite this sum as the sum over the orbits of S under G:

$$\sum_{s \in S} \frac{1}{|Gs|} = \sum_{i=1}^{m} \sum_{t \in G_{y_i}} \frac{1}{|Gt|} = \sum_{i=1}^{m} 1 = m$$

   which is the number of distinct orbits.

MA347 – HW17

Jonathan Lam

April 4, 2021

1. Let G be a group and H be a subgroup. Suppose for each $a, b \in G$ there exists $c \in G$ such that $aHbH = cH$. Prove that ${}^{G}/_{H}$, the set of left cosets of H in G, is a group.

   Proof. The group operation is a binary (closed) operation by definition, since for all $x, y \in G$, $(xH)(yH) = xHyH = cH \in {}^{G}/_{H}$ for some $c \in H$.

   GRP1 Product of (co)sets is always associative. Consider the product $((aH)(bH))(cH)$, where $a, b, c \in G$.

   $$\begin{aligned}((aH)(bH))(cH) &= \{(a'b')c' : a' \in aH, b' \in bH, c' \in cH\} \\ &= \{a'(b'c') : a' \in aH, b' \in bH, c' \in cH\} \\ &= (aH)((bH)(cH))\end{aligned}$$

   GRP2 Let $e' = H = eH \in {}^{G}/_{H}$, where e is the unit element of G. Then $\forall aH \in {}^{G}/_{H}$:

   $$(aH)(H) = a(HH) = aH$$

   Conversely, let $x \in H(aH)$. Then $x = h_1 a h_2$. In particular, let $h_1 = e \Rightarrow x \in aH$. We know that $H(aH)$ is a left coset of H in G. Since left cosets form an equivalence relation over G, $x \in aH \Rightarrow H(aH)$ is precisely the coset aH.

   $(aH)e' = aH = e'(aH) \therefore e' = H$ is the unit element of ${}^{G}/_{H}$.

   GRP3 Let $aH \in {}^{G}/_{H}$. Then $a^{-1}H \in {}^{G}/_{H}$. Then $(aH)(a^{-1}H)$ is a left coset of H in G. Let $x = ah_1a^{-1}h_2 \in (aH)(a^{-1}H)$. In particular, let $h_1 = e$. Then $x = aea^{-1}h_2 = aa^{-1}h_2 \in eH = H = e'$. Since left cosets form an equivalence relation over G, $x \in aH \Rightarrow (aH)(a^{-1}H)$ is precisely the coset $H = e'$.

   The same argument can be used to show that $(a^{-1}H)(aH) = H = e'$. Thus an inverse element exists for each $aH \in {}^{G}/_{H}$.

   This set of left cosets with the stated property is closed over the binary group operation and the group axioms are satisfied, so it is a group under product. □

2. Let G and G' be abelian groups and $f : G \to G'$ be a homomorphism. Assume there exists a homomorphism $g : G' \to G$ such that $f \circ g = I_{G'}$.

   (a) Prove that $G = \operatorname{Ker} f \oplus \operatorname{Im} g$.

   Proof. ($G = \operatorname{Ker} f + \operatorname{Im} g$) Let $x \in G$. Then $y = g(f(x)) \in \operatorname{Im} g$. Let $z = x - y$. Then

$$\begin{aligned} f(z) &= f(x-y) && \text{(def. } z\text{)} \\ &= f(x) - f(y) && (f \text{ homo.}) \\ &= f(x) - f(g(f(x))) && \text{(def. } y\text{)} \\ &= f(x) - f(x) && (f \circ g = I_{G'}) \\ &= e' \\ &\Rightarrow z \in \operatorname{Ker} f \end{aligned}$$

   Thus $x = z + y$ for $z \in \operatorname{Ker} f$, $y \in \operatorname{Im} g$. $\therefore G = \operatorname{Ker} f = \operatorname{Im} g$.

   ($\operatorname{Ker} f \cap \operatorname{Im} g = \{0\}$) Let $x \in \operatorname{Ker} f \cap \operatorname{Im} g$. Then $\exists y' \in G'$ such that $y = g(y') \Rightarrow f(y) = g(f(y')) = y' = 0' \Rightarrow y = g(y') = g(0') = 0$. Thus $\operatorname{Ker} f \cap \operatorname{Im} g = \{0\}$.

   $G = \operatorname{Ker} f + \operatorname{Im} g, \operatorname{Ker} f \cap \operatorname{Im} g = \{0\} \Rightarrow G = \operatorname{Ker} f \oplus \operatorname{Im} g$. □

   (b) Prove that f and g are inverse isomorphisms between the abelian groups $g(G')$ and G'.

   Proof. Restrict the domain of f and codomain of g to be the subgroup $g(G') \subseteq G$:

$$\begin{aligned} f &: g(G') \to G' \\ g &: G' \to g(G') \end{aligned}$$

   ($g \circ f = I_{g(G')}$) Let $x \in g(G')$. Then $x = g(x')$ for some $x' \in G'$, and

$$\begin{aligned} (g \circ f)(x) &= g(f(x)) && \text{(def. } \circ\text{)} \\ &= g(f(g(x'))) && (x \in g(G')) \\ &= g(x') && (f \circ g = I_{G'}) \\ &= x && (x = g(x')) \\ &= I_{g(G')}(x) && \text{(def. } I_{g(G')}\text{)} \end{aligned}$$

   We have $f : g(G') \to G'$, $g : G' \to g(G')$ homomorphisms, and $f \circ g = I_{G'}$ (given), $g \circ f = I_{g(G')}$ (just proved). $\therefore f$ and g are inverse isomorphisms between G' and $g(G')$. □

MA347 – HW18

Jonathan Lam

April 5, 2021

1. Find all abelian groups of order 1160 up to isomorphism.

$$1160 = (2^3)(5)(29)$$

There are three groups of order 8 up to isomorphism: $\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$, $\mathbb{Z}_{2^2} \oplus \mathbb{Z}_2$, and $\mathbb{Z}_{2^3}$. There is only one group of order 5 and one group of order 29 up to isomorphism: the cyclic groups $\mathbb{Z}_5$ and $\mathbb{Z}_{29}$, respectively.

Thus there are three groups with order 1160 up to isomorphism.

$$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_{29} \simeq \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_{290}$$
$$\mathbb{Z}_2 \oplus \mathbb{Z}_{2^2} \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_{29} \simeq \mathbb{Z}_2 \oplus \mathbb{Z}_{580}$$
$$\mathbb{Z}_{2^3} \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_{29} \simeq \mathbb{Z}_{1160}$$

[It is easy to see that any group of order 1160 falls into one of these sets (due to the factorization of 1160), and that these isomorphism classes are mutually distinct (due to the theorem from class).]

2. Find the invariant factors and the elementary divisors of:

(a) $A = \mathbb{Z}_{10} \oplus \mathbb{Z}_{20} \oplus \mathbb{Z}_{30} \oplus \mathbb{Z}_{40}$

$$\begin{aligned}
\mathbb{Z}_{10} &\simeq \mathbb{Z}_2 \oplus \mathbb{Z}_5 \\
\mathbb{Z}_{20} &\simeq \mathbb{Z}_{2^2} \oplus \mathbb{Z}_5 \\
\mathbb{Z}_{30} &\simeq \mathbb{Z}_2 \oplus \mathbb{Z}_3 \oplus \mathbb{Z}_5 \\
\mathbb{Z}_{40} &\simeq \mathbb{Z}_{2^3} \oplus \mathbb{Z}_5
\end{aligned}$$

thus

$$A \simeq \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_{2^2} \oplus \mathbb{Z}_{2^3} \oplus \mathbb{Z}_3 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_5$$

and $\{2, 2, 4, 8, 3, 5, 5, 5, 5\}$ are the elementary divisors of A. Using the visual method from class for invariant factors:

2	2	2^2	2^3
			3
5	5	5	5
10	10	20	120

which are the four invariant factors of A.

(b) $B = \mathbb{Z}_{12} \oplus \mathbb{Z}_{30} \oplus \mathbb{Z}_{100} \oplus \mathbb{Z}_{240}$

$$\begin{aligned}
\mathbb{Z}_{12} &\simeq \mathbb{Z}_{2^2} \oplus \mathbb{Z}_3 \\
\mathbb{Z}_{30} &\simeq \mathbb{Z}_2 \oplus \mathbb{Z}_3 \oplus \mathbb{Z}_5 \\
\mathbb{Z}_{100} &\simeq \mathbb{Z}_{2^2} \oplus \mathbb{Z}_{5^2} \\
\mathbb{Z}_{240} &\simeq \mathbb{Z}_{2^4} \oplus \mathbb{Z}_3 \oplus \mathbb{Z}_5
\end{aligned}$$

thus

$$B \simeq \mathbb{Z}_2 \oplus \mathbb{Z}_{2^2} \oplus \mathbb{Z}_{2^2} \oplus \mathbb{Z}_{2^4} \oplus \mathbb{Z}_3 \oplus \mathbb{Z}_3 \oplus \mathbb{Z}_3 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_{5^2}$$

and $\{2, 4, 4, 16, 3, 3, 3, 5, 5, 25\}$ are the elementary divisors of B.

2	2^2	2^2	2^4
	3	3	3
	5	5	5^2
2	60	60	1200

which are the four invariant factors of B.

MA347 – HW19

Jonathan Lam

April 8, 2021

1. Let A be an abelian group where $A = \mathbb{C}^*$ under product. Let p be a prime. Find $A(p)$, the p-primary component of A.

   By definition,

   $$\begin{aligned} A(p) &= \left\{ x \in \mathbb{C}^* : \exists n \in \mathbb{Z}^+ \text{ s.t. } x^{(p^n)} = 1 \right\} \\ &= \bigcup_{n=0}^{\infty} \left\{ x \in \mathbb{C}^* : x^{(p^n)} = 1 \right\} \\ &= \bigcup_{n=0}^{\infty} \left\{ \exp \frac{2\pi i k}{p^n} : 0 \le k < p^n \right\} \end{aligned}$$

   This is the set of the zeroth/first root of 1 (1 itself), p-roots of 1, p^2-roots of 1, p^3-roots of 1, etc. Since each p^n-th root of 1 is also a p^m-th root of 1 if $m > n$, this is the set of "p^∞-th roots of 1."

   (From a Google search, this group is called the Prüfer p-group $Z(p^\infty)$.)

2. Let G be a group and $H \triangleleft G$. Prove that:

   (a) If G is a p-group (for $p \in \mathbb{N}$ prime), then H is a p-group and G/H is a p-group.

   Proof. By definition, a p-group has (finite) order p^n, $n \in \mathbb{Z}^+$. Then $|G| = p^a$, $a \in \mathbb{Z}^+$. We have:

   $$|G| = (G : H)|H| = |G/H||H|$$

   By the Fundamental Theorem of Arithmetic, p^a can only be the product of the form $p^b p^c$, for $0 \le b, c \le a$ and $b + c = a$. Thus H and G/H (a group because $H \triangleleft G$) are p-groups. □

   (b) If H and G/H are p-groups, then G is a p-group.

   Proof. H is a p-group $\Rightarrow |H| = p^c$ for $c \in \mathbb{Z}^+$. G/H is a p-group $\Rightarrow |G/H| = (G : H) = p^b$ for $b \in \mathbb{Z}^+$. Then

   $$|G| = (G : H)|H| = p^b p^c = p^{b+c}$$

   The order of G is a power of p, so G is a p-group. □

MA347 – HW20

Jonathan Lam

April 18, 2021

1. Prove that an abelian group A is simple iff it is finite and of prime order.

   Proof. Preliminaries: since A is abelian, every subgroup is normal. If it is simple, every nonzero (non-identity) element $a \in A$ generates a nontrivial subgroup of A, which must be A itself (every nonzero element must be a generator of A).

   ($\Rightarrow$) Assume that an abelian group A is simple and of infinite order. Let $0 \neq a \in A$, and thus a generates A. The nonzero element $2a$ must also generate A, and thus there exists $m \in \mathbb{N}$ such that $m(2a) = a$. Subtracting a from both sides, we have $(2m-1)a = 0$, and thus a has a finite exponent $\Rightarrow A = \langle a \rangle$ is finite $\Rightarrow$ contradiction. $\therefore A$ is simple $\Rightarrow A$ is finite.

   Let A be a finite abelian group with non-prime order, and let p be a prime that divides $|A|$. By Lemma 9.2 (Lang), A has a (proper) subgroup of order p, so A is not simple. By contrapositive, A must have prime order.

   ($\Leftarrow$) By Lagrange's theorem, any subgroup of a (finite) group A must have order dividing A. A has prime order, so any (normal) subgroup must be the trivial group or itself. $\therefore A$ is simple. □

2. Prove that S_4 has more than one 2-Sylow subgroup.

   Proof. By Sylow's third theorem we know that S_4 has either one or three 2-sylow subgroups (subgroups of order 8).

   Consider S_4 as the set of permutations of the integers J_4. Choose the subgroup of S_4 that maintains the "partition" of S_4 $\{1,2\} \cup \{3,4\}$, i.e., the permutations that map $\{1,2\}$ onto either $\{1,2\}$ or $\{3,4\}$:

   $$\{\varepsilon, (12), (34), (12)(34), (13)(24), (14)(23), (1324), (1423)\} = H < S_4$$

   This can be shown to be a subgroup (almost by definition). These are all the possibilities that preserve this partition; for $\sigma \in H$, $\sigma(1)$ has four possibilities, $\sigma(2)$ is fixed by $\sigma(1)$, $\sigma(3)$ has two possibilities (limited by the choice of $\sigma(1)$), and $\sigma(4)$ is fixed by the other choices, so $|H| = 8$.

   There are three unique partitions of S_4 into two sets of two elements, so these must be the three 2-Sylow subgroups of S_4. □

MA347 – HW21

Jonathan Lam

April 18, 2021

1. Let G be a group of order 91.

   (a) Prove that G has only one 7-Sylow subgroup H and only one 13-Sylow subgroup K.

   Proof. Using Sylow's third theorem, there are $m = 1 + 7k$ 7-Sylow subgroups, $m \mid 91$, and there are $n = 1 + 13k$ 13-Sylow subgroups, $n \mid 91$. The only numbers that match these constraints are $m = 1$ and $n = 1$ ($k = 0$ in both cases). □

   (b) Prove that H, K are normal.

   Proof. Let G be a group, and P be the only p-Sylow subgroup. Let $g \in G$. Then the conjugate group gPg^{-1} also is a p-Sylow subgroup ($|gPg^{-1}| = |P|$ since conjugation is a bijection), and must equal P because it is the only p-Sylow subgroup. Thus $gPg^{-1} = P$ for all $g \in G$, so P is normal. (This proof is true more generally for the case of a unique subgroup of any order.)
   Apply this result to H and K to show that they are normal. □

   (c) Prove that $H \cap K = \{e\}$.

   Proof. H and K are both finite groups of prime order, so they are cyclic groups where each non-identity element has period matching the order of the subgroup. Thus no non-identity element can belong to both subgroups (no element can have period both 7 and 13), so the intersection is trivial. □

(d) Prove that $hk = kh \ \forall h \in H, k \in K$, and that H, K are abelian.

Proof. $(hk = kh \ \forall h \in H, k \in K)$
Consider the element $hkh^{-1}k^{-1}$, $h \in H$, $k \in K$. $K \triangleleft G \Rightarrow hkh^{-1} = k_1 \in K$. $H \triangleleft G \Rightarrow kh^{-1}k^{-1} = h_1 \in H$. Then

$$kk_1 = hkh^{-1}k^{-1} = hh_1 \Rightarrow hkh^{-1}k^{-1} \in H \cap K = \{e\}$$

$\therefore hk = kh$.
(H, K abelian) H and K are both finite groups of prime order. Thus it must be a (simple) (cyclic) abelian group (previous HW). □

(e) Prove that G is cyclic and hence abelian.

Proof. Suppose $a = hk \in G$, where $h \in H$, $k \in K$, and $h, k \neq e$. Then h has period 7 and k has period 13. Since H and K are abelian, and since elements of H and K commute with each other, we can write $a^n = (hk)^n = h^n k^n$ (proof using induction is immediate).
We show that 91 is the period of a. It is an exponent of a because $a^{91} = h^{91}k^{91} = (h^7)^{13}(k^{13})^7 = e^{13}e^7 = ee = e$. It is the least positive exponent because for any $0 < n < 91$, 7 and 13 do not both divide n, so h^n and k^n are not both e (nor are they inverses because $H \cap K = \{e\}$), and thus $a^n = h^n k^n \neq e$.
a generates a subgroup of G of order 91, so $G = \langle a \rangle$ and thus abelian. □

MA347 – HW22

Jonathan Lam

April 26, 2021

1. Prove that $\mathbb{Q}[\sqrt{2}] = \{a + \sqrt{2}b : a, b \in \mathbb{Q}\}$ is an integral domain.

   To show this, we must first show that $R = \mathbb{Q}[\sqrt{2}]$ is a ring, and then that it is a commutative ring with identity, and then that it has no zero divisors.

   Proof. (R is a ring.) Let $a = a_1 + \sqrt{2}a_2, b = b_1 + \sqrt{2}b_2 \in R$, for $a_1, a_2, b_1, b_2 \in \mathbb{Q}$. Define the sum element-wise. Define the product by

   $$\cdot(a, b) = ab = (a_1b_1 + 2a_2b_2) + \sqrt{2}(a_1b_2 + a_2b_1)$$

   Due to the closure of $\mathbb{Q}$ over product, $(a_1b_1 + 2a_2b_2), (a_1b_2 + a_2b_1) \in R$, so $ab \in R$.

   R1 $(R, +)$ is an abelian group due to the commutativity of the addition of rational numbers. (Proof not shown here b/c trivial.)

   R2 Let $a = a_1+\sqrt{2}a_2, b = b_1+\sqrt{2}b_2, c = c_1+\sqrt{2}c_2 \in R$, for $a_1, a_2, b_1, b_2, c_1, c_2 \in \mathbb{Q}$. Then:

   $$\begin{aligned}
   (ab)c &= ((a_1b_1 + 2a_2b_2) + \sqrt{2}(a_1b_2 + a_2b_1))c \\
   &= ((a_1b_1 + 2a_2b_2)(c_1) + 2(a_1b_2 + a_2b_1)(c_2)) \\
   &\quad + \sqrt{2}((a_1b_1 + 2a_2b_2)(c_2) + (a_1b_2 + a_2b_1)(c_1)) \\
   &= (a_1(b_1c_1 + 2b_2c_2) + 2a_2(b_1c_2 + b_2c_1)) \\
   &\quad + \sqrt{2}(a_1(b_1c_2 + b_2c_1) + a_2(b_1c_1 + 2b_2c_2)) \\
   &= a((b_1c_1 + 2b_2c_2) + \sqrt{2}(b_1c_2 + b_2c_1)) \\
   &= a(bc)
   \end{aligned}$$

   $\therefore$ product is associative.

R3 Let $a, b, c \in R$ as above. Then:

$$\begin{aligned} a(b+c) &= a((b_1+c_1)+\sqrt{2}(b_2+c_2)) \\ &= (a_1(b_1+c_1)+2a_2(b_2+c_2)) \\ &\quad + \sqrt{2}(a_1(b_2+c_2)+a_2(b_1+c_1)) \\ &= ((a_1b_1+2a_2b_2)+(a_1c_1+2a_2c_2)) \\ &\quad + \sqrt{2}((a_1b_2+a_2b_1)+(a_1c_2+a_2c_1)) \\ &= ((a_1b_1+2a_2b_2)+\sqrt{2}(a_1b_2+a_2b_1)) \\ &\quad + ((a_1c_2+a_2c_1)+\sqrt{2}(a_1c_2+a_2c_1)) \\ &= ab+ac \end{aligned}$$

(The proof of right distributivity follows likewise due to the commutativity and distributivity of $\mathbb{Q}$.)

$\therefore$ addition distributes over product.

R1, **R2**, **R3** are satisfied $\therefore$ R is a ring.

(Show that R is commutative and unital.) Use the commutativity of sum and product in $\mathbb{Q}$ to show that product in R is commutative. Assume $a, b \in R$ as previously:

$$\begin{aligned} ab &= (a_1b_1+2a_2b_2)+\sqrt{2}(a_1b_2+a_2b_1) \\ &= (b_1a_1+2b_2a_2)+\sqrt{2}(b_2a_1+b_1a_2) \\ &= (b_1a_1+2b_2a_2)+\sqrt{2}(b_1a_2+b_2a_1) \\ &= ba \end{aligned}$$

R has identity $e = 1+\sqrt{2}(0)$, because $\forall a \in R$:

$$ea = (1a_1+2(0)a_2)+\sqrt{2}(1a_2+0a_1) = a_1+\sqrt{2}a_2 = a$$

and $ea = a = ae$ because R is commutative.

(Show that R has no zero divisors.) Assume $ab = 0 = 0+\sqrt{2}(0)$ for some $a, b \in R$ and a nonzero.

$$(a_1b_1+2a_2b_2)+\sqrt{2}(a_1b_2+a_2b_1) = 0+\sqrt{2}(0) \Rightarrow \begin{cases} a_1b_1+2a_2b_2 = 0 \\ a_1b_2+a_2b_1 = 0 \end{cases}$$

$a \neq 0 \Rightarrow a_1, a_2$ are not both zero. Assume $a_1, b_1, b_2 \neq 0$ (i.e., $b \neq 0$). Then:

$$b_1 = -\frac{2a_2b_2}{a_1}, \quad b_2 = -\frac{a_2b_1}{a_1}$$

$$\Rightarrow b_1 = \frac{2a_2^2}{a_1^2}b_1 \Rightarrow \frac{2a_2^2}{a_1^2} = 1 \Rightarrow \frac{a_2}{a_1} = \frac{1}{\sqrt{2}} \Rightarrow a_2 = \frac{1}{\sqrt{2}}a_1$$

which is a contradiction since $a_2 = \frac{1}{\sqrt{2}} a_1 \notin \mathbb{Q}$. Thus $a_1 \neq 0 \Rightarrow b = 0$.

Similarly, if we assume that $a_2, b_1, b_2 \neq 0$ (i.e., $b \neq 0$), then:

$$b_2 = -\frac{a_1 b_1}{2a_2}, \quad b_1 = -\frac{a_1 b_2}{a_2}$$
$$\Rightarrow b_2 = \frac{a_1^2}{2a_2^2} b_2 \Rightarrow \frac{a_1^2}{2a_2^2} = 1 \Rightarrow \frac{a_1}{a_2} = \sqrt{2} \Rightarrow a_1 = \sqrt{2} a_2$$

which is again a contradiction because $\sqrt{2} \notin \mathbb{Q}$.

$\therefore ab = 0, a \neq 0 \Rightarrow b = 0$. (Similarly, $b \neq 0 \Rightarrow a = 0$ because R is commutative.) Thus R has no zero divisors. $\square$

2. Let R be a commutative ring with identity. Let L, M, and N be (two-sided) ideals. Prove that:

   (a) $M + N$ is a left ideal of R.

   Proof. Let $x = m_1 + n_1 \in M + N$ and $r \in R$, for $m_1, m_2 \in M, n_1, n_2 \in N$. Then:

$$\begin{aligned} x + y &= (m_1 + n_1) + (m_2 + n_2) \\ &= (m_1 + m_2) + (n_1 + n_2) \\ &= m_3 + n_3 \in M + N \end{aligned}$$

   for $m_3 \in M, n_3 \in N$, since M, N are ideals and thus closed over addition. $\therefore M + N$ is closed over addition. We also have:

$$\begin{aligned} rx &= r(m_1 + n_1) \\ &= rm_1 + rn_2 \\ &= m_4 + n_4 \in M + N \end{aligned}$$

   for $m_4 \in M, n_4 \in N$, since M, N are ideals and closed over product with an element of R. $\therefore M + N$ is closed over product with an element of r.
   $\therefore M + N$ is a left ideal. (It is also a right ideal by the symmetric argument.) $\square$

   (b) $L(M + N) = LM + LN$

   Proof. Let $l \in L, m \in M, n \in N$. By definition of product of ideals:

$$\begin{aligned} LM + LN &= \left\{ \sum_{i=1}^{s} l_i m_i : l_i \in L, m_i \in M, s \in \mathbb{Z}^+ \right\} \\ &\quad + \left\{ \sum_{i=1}^{t} l_i n_i : l_i \in L, n_i \in N, t \in \mathbb{Z}^+ \right\} \\ &= \left\{ \sum_{i=1}^{u} l_i m_i + l_i n_i : l_i \in L, m_i \in M, n_i \in N, u \in \mathbb{Z}^+ \right\} \\ &= \left\{ \sum_{i=1}^{u} l_i (m_i + n_i) : l_i \in L, m_i \in M, n_i \in N, u \in \mathbb{Z}^+ \right\} \\ &= L(M + N) \end{aligned}$$

   (Note that when we move from the two summations with upper limits s and t to the single summation with upper limit u, we "fill in" the missing terms with 0's, since $0 \in M, N$.) $\square$

(c) $LM \subseteq L \cap M$

Proof. Let $x = lm \in LM$, $l \in L, m \in M$. By definition of the product of ideals:

$$LM = \left\{ \sum_{i=1}^{n} l_i m_i : l_i \in L, m_i \in M, n \in \mathbb{Z}^+ \right\}$$

Since L is a right ideal, each term $l_i m_i \in L$, and the linear combination lies in L. Similarly, since M is a left ideal, each term $l_i m_i \in M$, and the linear combination lies in M. Thus the linear combination lies in $L \cap M$, so $LM \subseteq L \cap M$. □

MA347 – HW23

Jonathan Lam

April 28, 2021

1. (a) Let I be an ideal of a commutative ring R. Prove that $\text{ann}(I) = \{r \in R : ra = 0 \ \forall a \in I\}$ is an ideal of R.

   Proof. Let $x_1, x_2 \in \text{ann}(I)$, $r \in R$. Then, $\forall a \in I$:

   $$a(x_1 + x_2) = ax_1 + ax_2 = 0 + 0 = 0$$

   and

   $$a(rx_1) = r(ax_1) = r(0) = 0$$

   $\therefore x_1 + x_2 \in \text{ann}(I), rx_1 = x_1 r \in \text{ann}(I) \Rightarrow \text{ann}(I)$ is a (two-sided) ideal. □

   (b) Let $R = \mathbb{Z}/20\mathbb{Z}$ and $I = \{[n] \in R : n \text{ is even}\}$. Find $\text{ann}(I)$.
   We use the looser notation $r \equiv [r] \in R$.

   $$\begin{aligned}
   \text{ann}(I) &= \{r \in \mathbb{Z}/20\mathbb{Z} : ra = 0 \ \forall a \in I\} \\
   &= \{r \in \mathbb{Z}/20\mathbb{Z} : ra = 0 \ \forall a \in \{0, 2, \ldots, 18\}\} \\
   &= \{r \in \mathbb{Z}/20\mathbb{Z} : ra \equiv 0 \pmod{20} \ \forall a \in \{0, 2, \ldots, 18\}\} \\
   &= \{r \in \mathbb{Z}/20\mathbb{Z} : 20 \mid ra \ \forall a \in \{0, 2, \ldots, 18\}\}
   \end{aligned}$$

   Then $\text{ann}(I)$ must be the ideal $\{0, 10\}$, because no other elements have exponent 2 in the additive group, and clearly $0a \equiv 0 \pmod{20}$, $10a \equiv 0 \pmod{20}$ $\forall a \in \{0, 2, \ldots, 18\}$.

2. Find all the ideals of a field F.

   The trivial ideal $\{0\} \subseteq F$ is always an ideal.

   Assume an ideal $J \subseteq F$ is non trivial. Let $0 \neq j \in J$. Since F^* is a group under product, $\exists r = j^{-1} \in R$. Since J is an ideal, $1 = j^{-1}j = rj \in R$. Then $\forall r \in R$, $r(1) \in J \Rightarrow R \subseteq J \subseteq R \Rightarrow J = R$.

   $\therefore$ the only two ideals of F are the trivial ideal and F itself.

MA347 – HW24

Jonathan Lam

May 5, 2021

1. Let R be a ring and $I \subseteq J \subseteq R$ where I and J are two-sided ideals. Prove that there is a (unique) ring homo. $\varphi : {}^{R}/_{I} \to {}^{R}/_{J}$ such that $\varphi(a+I) = a+J$.

   Proof. First, we should show that φ is indeed a ring homo.

   Let $a+I, b+I \in {}^{R}/_{I}$, where $a, b \in R$. The product and sum of these factor rings are well defined:

   $$\begin{aligned}
   (a+I)(b+I) &= a(b+I) + I(b+I) && \text{(rings are distributive)} \\
   &= ab + aI + bI + II && \text{(rings are distributive)} \\
   &= ab + I + I + II && (I \text{ is two-sided ideal}) \\
   &= ab + I && (II = I = I + I) \\
   (a+I) + (b+I) &= a + b + I + I && \text{(additive group is abelian)} \\
   &= (a+b) + I
   \end{aligned}$$

   The fact that φ is a ring homomorphism follows naturally:

   $$\begin{aligned}
   \varphi((a+I)(b+I)) &= f(ab+I) \\
   &= ab + J \\
   &= (a+J)(b+J) \\
   &= \varphi(a+I)\varphi(b+I) \\
   \varphi((a+I) + (b+I)) &= \varphi((a+b)+I) \\
   &= (a+b) + J \\
   &= (a+J) + (b+J) \\
   &= \varphi(a+I) + \varphi(b+I) \\
   \varphi(e) &= \varphi(0+I) \\
   &= 0 + J = e'
   \end{aligned}$$

   $\therefore \varphi$ is a ring homo.

   We should also show that φ is well-defined. Let $a+I = b+I$ for $a, b \in R$. Then $a - b \in I \subseteq J \Rightarrow a + J = b + J$. Thus φ is well-defined.

   This map is unique because if $\psi : {}^{R}/_{I} \to {}^{R}/_{J}$ is a ring homomorphism that maps $a + I \mapsto a + J$ for all $a + I \in {}^{R}/_{I}$, then $\psi = \varphi$. $\square$

2. Let R be an integral domain and $f : R \to R$ is a ring automorphism. Prove that there is a unique automorphism $f^* : K \to K$ of fields such that $f^*(r) = f(r)\ \forall r \in R$ where K is the quotient field of R.

Proof. This proof is identical to the textbook example on pages 103-4 of Lang (with some additional commentary), except that f is an automorphism rather than an embedding.

To show uniqueness, let $a \neq 0 \in R$. If f^* is a homo., then we must have:

$$1 = f^*(1) = f^*\left(\frac{1}{a}\frac{a}{1}\right) = f^*\left(\frac{1}{a}\right) f^*(a)$$
$$\Rightarrow f^*\left(\frac{1}{a}\right) = [f^*(a)]^{-1} = \frac{1}{f^*(a)}$$

Now, consider the fact that $f^*(r) = f(r)\ \forall r \in R$, i.e., that f^* extends f to its fraction field. Thus $\forall a, b \in R$, $b \neq 0$ we must have:

$$f^*\left(\frac{a}{b}\right) = f^*\left(\frac{a}{1}\frac{1}{b}\right) = f^*(a)\,\frac{1}{f^*(b)} = \frac{f^*(a)}{f^*(b)} = \frac{f(a)}{f(b)}$$

Thus the map f^* is uniquely determined by the effect of the map f on R. Also, it is clear that f^* is an extension of f: $\forall a = a/1 \in R \subseteq K$, then $f^*(a/1) = f(a)/f(1) = f(a)$.

To show that f^* is well-defined, let $x = a/b, y = c/d \in K$ for $a, b, c, d \in R$, and $x = y$. Then

$$f^*(x) = \frac{f(a)}{f(b)}, \qquad f^*(y) = \frac{f(c)}{f(d)}$$
$$x = y \Rightarrow ad = bc$$
$$f(a)f(d) = f(ad) = f(bc) = f(b)f(c) \Rightarrow f^*(x) = f^*(y)$$

Lastly, we need to show that f^* is indeed an automorphism, which was assumed up till now. Let $a, c \in R$, $b, d \in R^*$:

$$\begin{aligned}
f^*\left(\frac{a}{b}\frac{c}{d}\right) = f^*\left(\frac{ac}{bd}\right) &= \frac{f(ac)}{f(bd)} = \frac{f(a)f(c)}{f(b)f(d)} \\
&= \frac{f(a)}{f(b)}\frac{f(c)}{f(d)} f^*\left(\frac{a}{b}\right) = f^*\left(\frac{c}{d}\right) \\
f^*\left(\frac{a}{b} + \frac{c}{d}\right) = f^*\left(\frac{ad+bc}{bd}\right) &= \frac{f(ad+bc)}{f(bd)} \\
&= \frac{f(a)f(d) + f(b)f(c)}{f(b) + f(d)} \\
&= \frac{f(a)}{f(b)} + \frac{f(c)}{f(d)} = f^*\left(\frac{a}{b}\right) + f^*\left(\frac{c}{d}\right) \\
f^*\left(\frac{1}{1}\right) = \frac{f(1)}{f(1)} &= \frac{1}{1} = 1
\end{aligned}$$

$\therefore f^*$ is a homo.

To show that f is a ring auto., define a map $g^* : K \to K$ by $a/b \mapsto f^{-1}(a)/f^{-1}(b)$. Then, for $\forall a, c \in R$ and $b, d \in R^*$:

$$g^*\left(f^*\left(\frac{a}{b}\right)\right) = g^*\left(\frac{f(a)}{f(b)}\right) = \frac{f^{-1}(f(a))}{f^{-1}(f(b))} = \frac{a}{b}$$
$$f^*\left(g^*\left(\frac{a}{b}\right)\right) = f^*\left(\frac{f^{-1}(a)}{f^{-1}(b)}\right) = \frac{f(f^{-1}(a))}{f(f^{-1}(b))} = \frac{a}{b}$$

Thus $g^* \circ f^* = I_K = f^* \circ g^* \Rightarrow f^*$ is a bijective ring homomorphism from K to itself $\therefore$ it is a ring isomorphism. $\square$