IS 609 - Homework #6

J. Hamski

March 10, 2016

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

1

Minimize: $cost(h, t, b, p) = 1.8h + 3.5t + 0.4b + 1.0p$

Subject to:

$0.5h + t + 2.0b + 6.0p \geq 40.0$

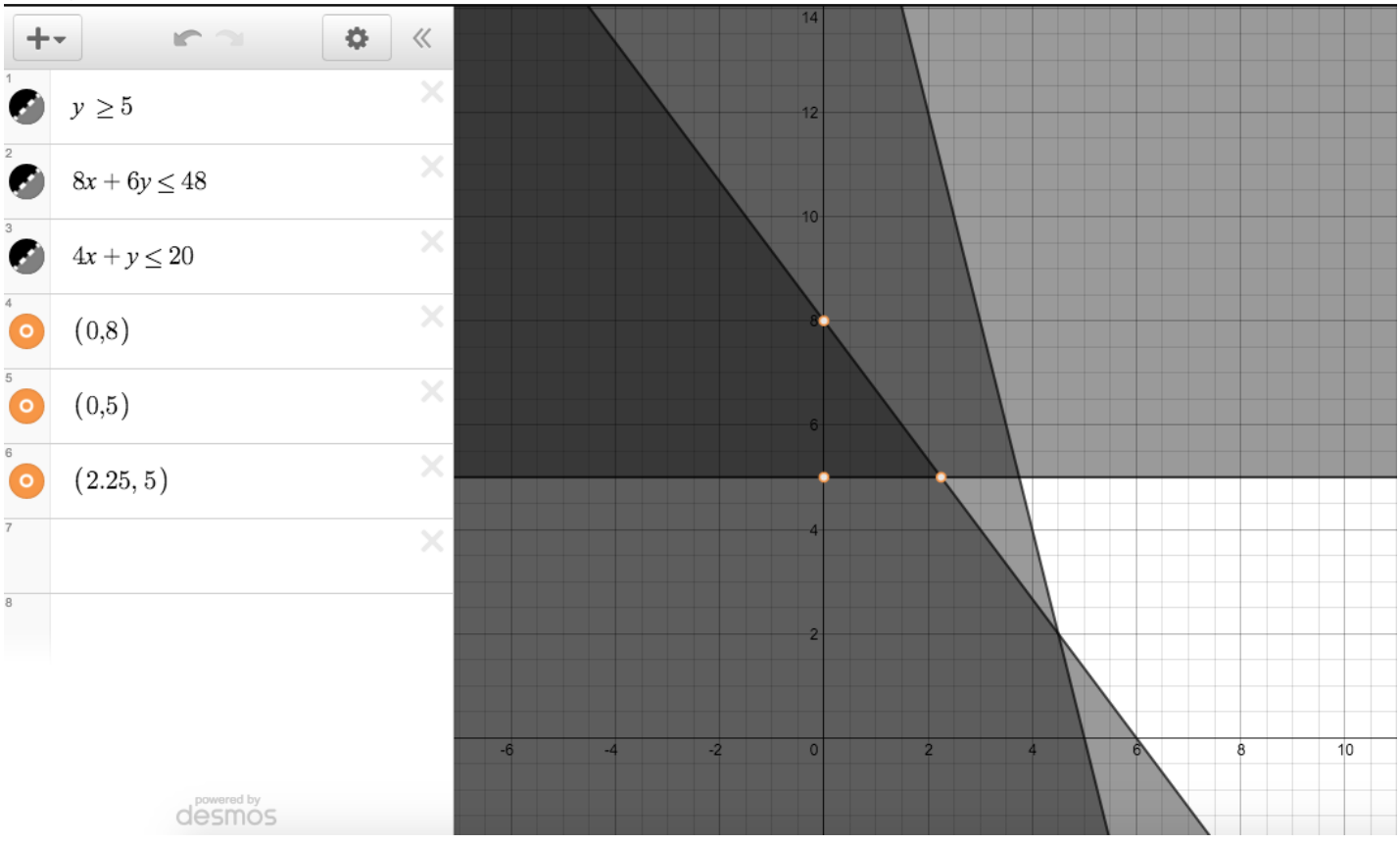
$2.0h + 4.0t + 0.5b + p \geq 20.0$

$5.0h + 2.0t + b + 2.5p \geq 25.0$

$h, t, b, p >= 0$

2

Using Desmos:

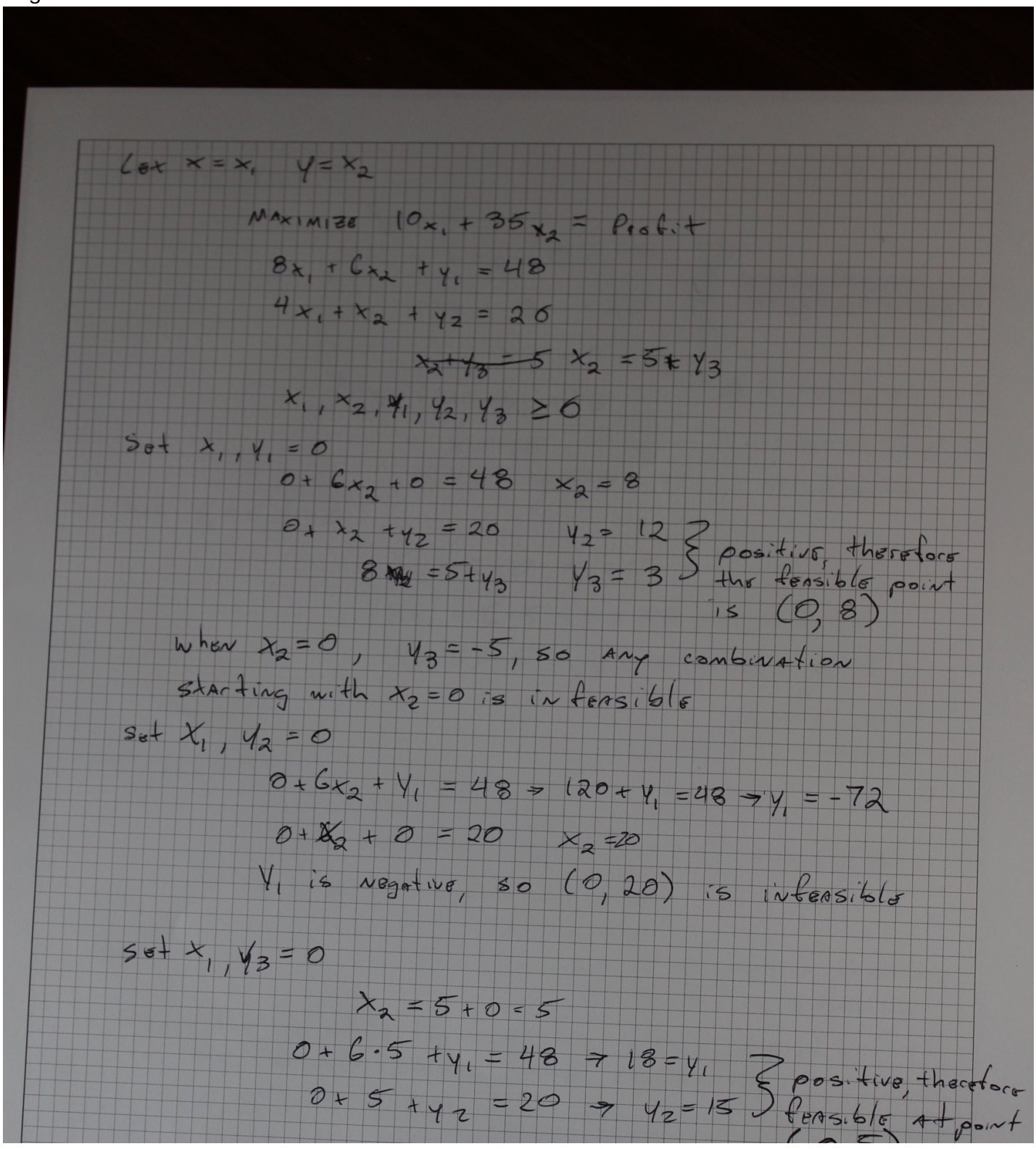


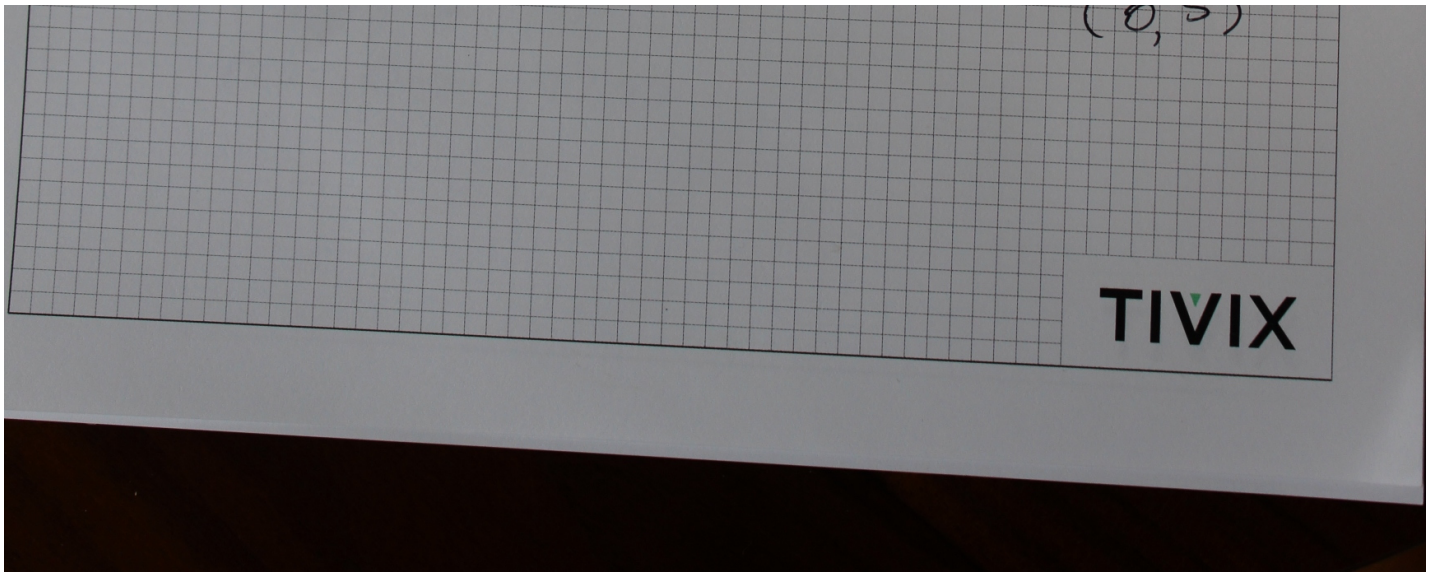
Extreme points and their objective function value:
x = 0, y = 8 -> **280**
x = 0, y = 5 -> 175
x = 2.25, y = 5 -> 197.50

280 is the maximum value

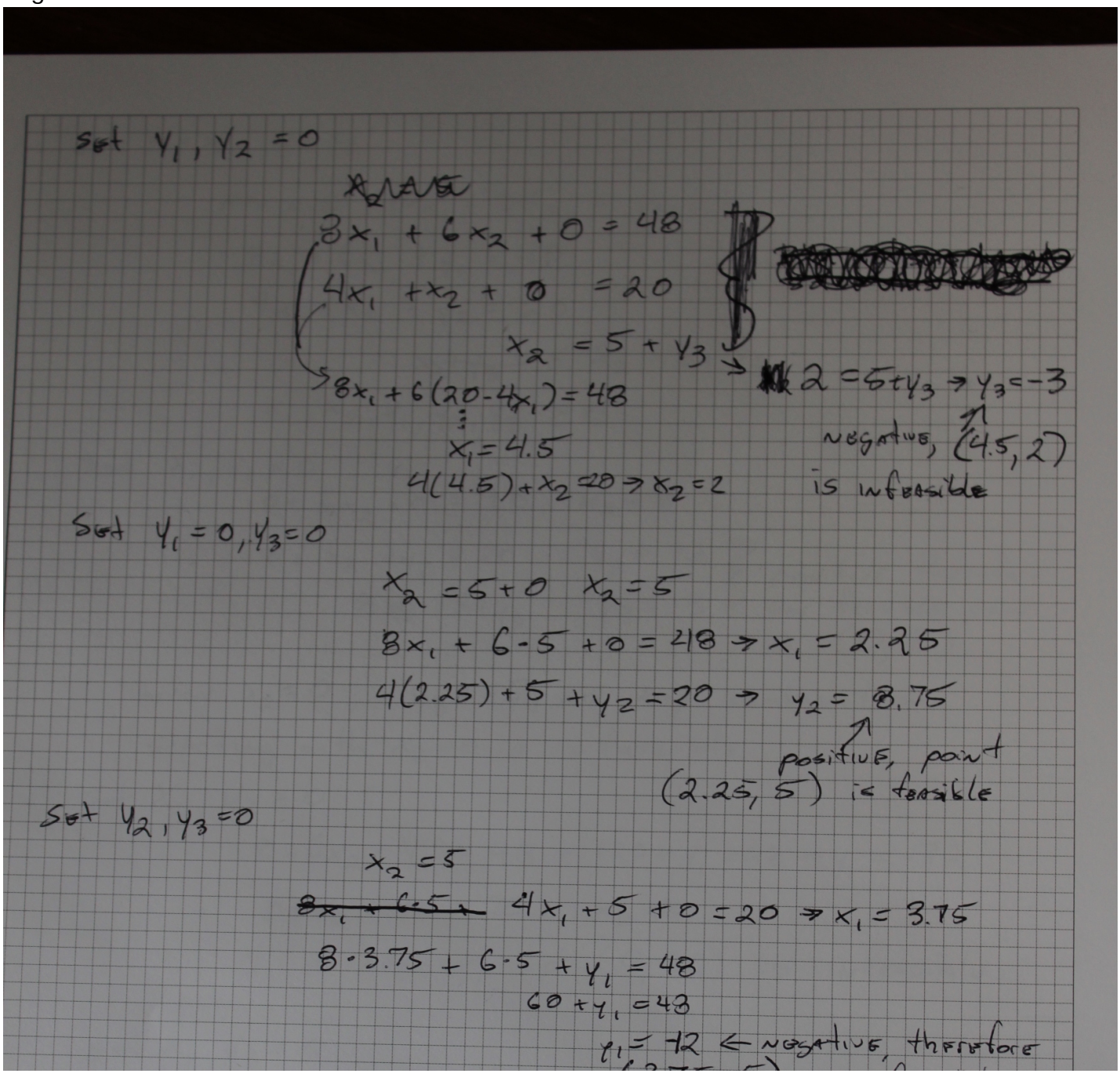
3

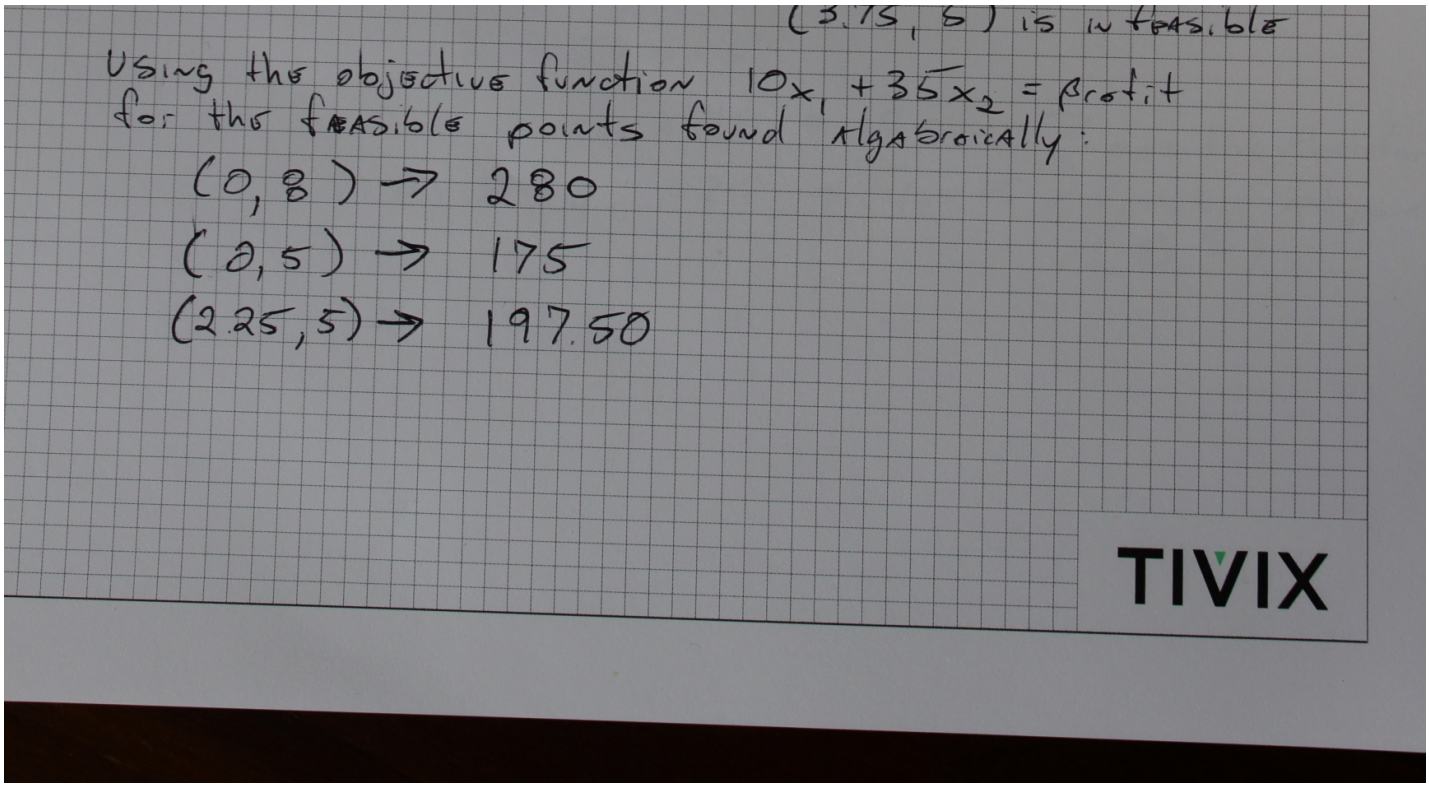
Page 1:




(0, 5)
TIVIX

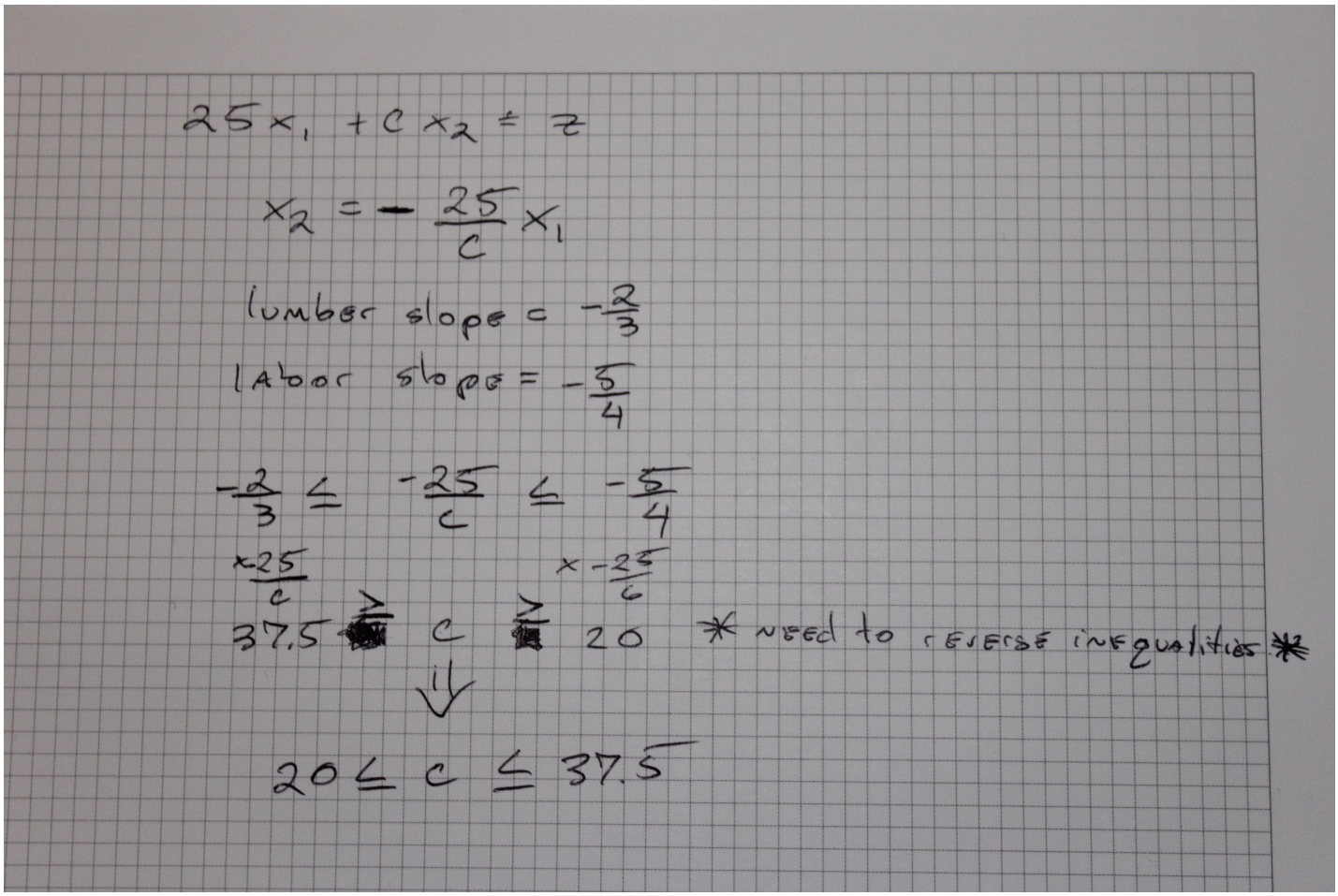
Page 2:


Set $Y_1, Y_2 = 0$
$3x_1 + 6x_2 + 0 = 48$
$4x_1 + x_2 + 0 = 20$
$x_2 = 5 + y_3$
$8x_1 + 6(20 - 4x_1) = 48$
$x_1 = 4.5$
$4(4.5) + x_2 = 20 \Rightarrow x_2 = 2$
$2 = 5 + y_3 \Rightarrow y_3 = -3$
negative, (4.5, 2) is infeasible
Set $y_1 = 0, y_3 = 0$
$x_2 = 5 + 0 \quad x_2 = 5$
$8x_1 + 6 \cdot 5 + 0 = 48 \Rightarrow x_1 = 2.25$
$4(2.25) + 5 + y_2 = 20 \Rightarrow y_2 = 8.75$
positive, point (2.25, 5) is feasible
Set $y_2, y_3 = 0$
$x_2 = 5$
$4x_1 + 5 + 0 = 20 \Rightarrow x_1 = 3.75$
$8 \cdot 3.75 + 6 \cdot 5 + y_1 = 48$
$60 + y_1 = 48$
$y_1 = -12$ ← negative, therefore


(3.75, 5) is infeasible
Using the objective function $10x_1 + 35x_2 = \text{profit}$
for the feasible points found algebraically:
(0, 8) → 280
(0, 5) → 175
(2.25, 5) → 197.50
TIVIX

4

5


$25x_1 + Cx_2 = Z$
$x_2 = -\frac{25}{C}x_1$
lumber slope = $-\frac{2}{3}$
labor slope = $-\frac{5}{4}$
$-\frac{2}{3} \leq -\frac{25}{C} \leq -\frac{5}{4}$
$\times \frac{25}{C}$ $\times -\frac{25}{C}$
$37.5 \geq C \geq 20$ * need to reverse inequalities *
$20 \leq C \leq 37.5$

6

```
x <- c(7, 14, 21, 28, 35, 42)
y <- c(6, 41, 133, 250, 280, 297)
```

The function to minimize is:

$$f(c) =| 6 - 7c | + | 41 - 14c | + | 133 - 21c | + | 250 - 28c | + | 280 - 31c | + | 297 - 42c|$$

on a closed interval $[0, 42]$.

a

$$y = ax$$

```
x <- c(7, 14, 21, 28, 35, 42)


t = 0.2

iterations.estimate <- log(t / (42)) / log(0.618)
iter <- round(iterations.estimate, 0)
iter
```

```
## [1] 11
```

```
error.model <- function(c){

  f.c <- abs(6 - 7 * c) + abs(41 - 14  * c) + abs(133 - 21 * c) + abs(250 - 28 *
c) + abs(280 - 31 * c) + abs(297 - 42 * c)
  return(f.c)

}
```

```
r = 0.618

a = 42
b = 0

x1 <- a + (1 - r) * (b - a)
x2 <- a + r * (b - a)

error.model(x1)
```

```
## [1] 2704.708
```

```
error.model(x2)
```

```
## [1] 1287.292
```

```
golden.endpoints <- function(endpoints){
  x1 <- endpoints[1] + (1 - r) * (endpoints[2] - endpoints[1])
  x2 <- endpoints[1] + r * (endpoints[2] - endpoints[1])
  endpoints <- c(x1, x2)
  return(endpoints)
}
```

```
test.endpoints <-function(endpoints, old.endpoints){
  f.x1 <- error.model(endpoints[1])
  f.x2 <- error.model(endpoints[2])

  if(f.x1 > f.x2){new.endpoints <- matrix(c(endpoints[1], old.endpoints[2]), nrow=
1, byrow=T)} else {new.endpoints <- matrix(c(old.endpoints[1], endpoints[2]), nrow
=1, byrow=T)}

  return(new.endpoints)
}
```

```
list.endpoints <- matrix(c(0, 42), nrow = 1, byrow=T)

for (i in 1:iter) {
  new.endpoints <- golden.endpoints(list.endpoints[i,])
  new.endpoints <- test.endpoints(new.endpoints, list.endpoints[i,])
  list.endpoints <- rbind(list.endpoints, new.endpoints)
}
list.endpoints
```

```
##           [,1]      [,2]
##  [1,] 0.000000 42.000000
##  [2,] 0.000000 25.956000
##  [3,] 0.000000 16.040808
##  [4,] 0.000000  9.913219
##  [5,] 3.786850  9.913219
##  [6,] 6.127123  9.913219
##  [7,] 6.127123  8.466931
##  [8,] 6.127123  7.573124
##  [9,] 6.679495  7.573124
## [10,] 6.679495  7.231758
## [11,] 6.890460  7.231758
## [12,] 7.020836  7.231758
```

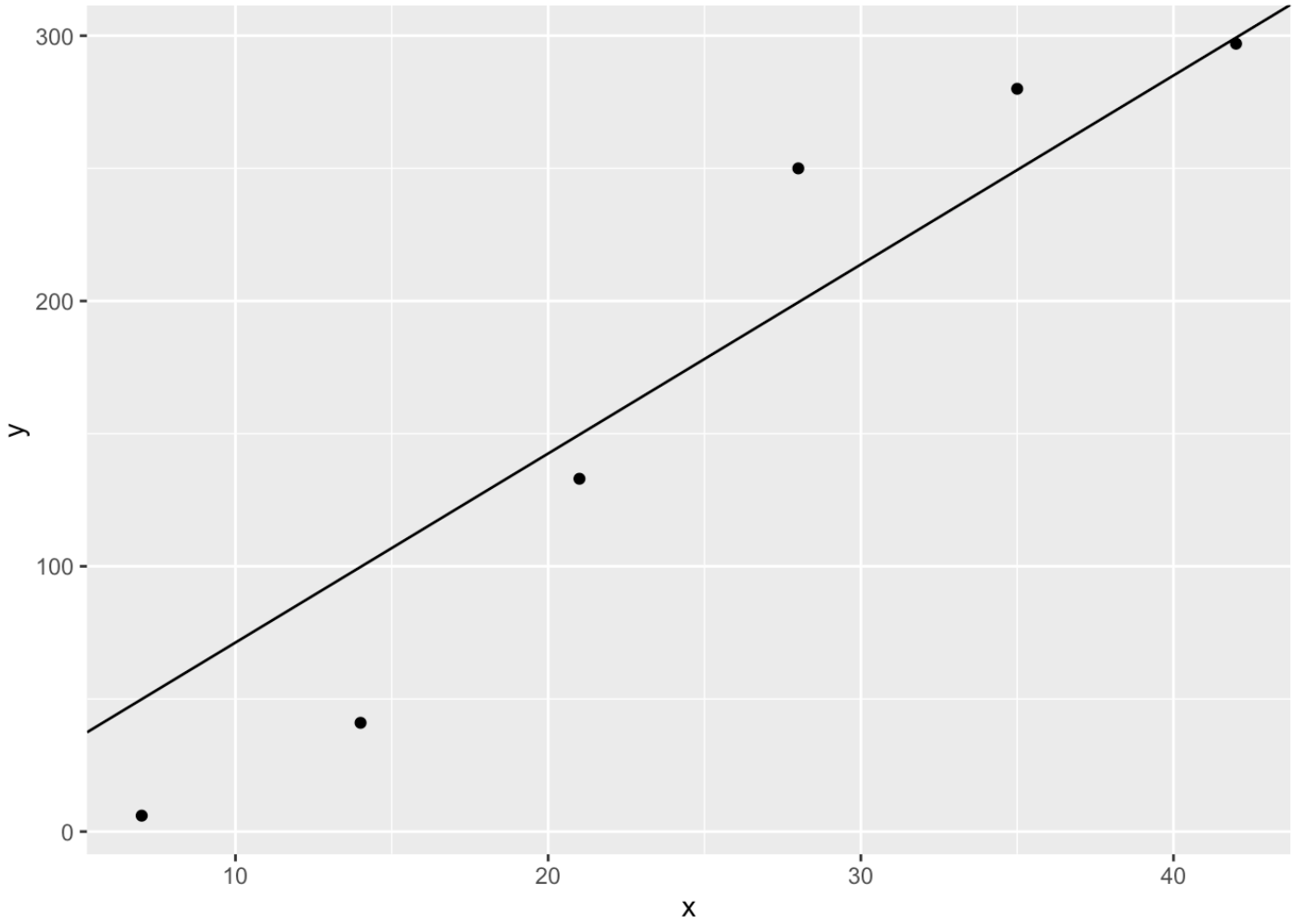
```
slope <- mean(list.endpoints[iter+1,])
slope
```

```
## [1] 7.126297
```

This isn't exactly what is in the answer section of the book. I think the difference is due to float point math precision differences.

```
data <- as.data.frame(cbind(x, y))

ggplot(data, aes(x=x, y=y)) + geom_point() + geom_abline(slope = slope, intercept
= 0)
```



b

$$y = ax^2$$

```
x <- c(7, 14, 21, 28, 35, 42) ** 2


t = 0.2

iterations.estimate <- log(t / (1764)) / log(0.618)
iter <- round(iterations.estimate, 0)
iter
```

```
## [1] 19
```

```
error.model <- function(c){

  f.c <- abs(6 - 49 * c) + abs(41 - 196  * c) + abs(133 - 441 * c) + abs(250 - 784
* c) + abs(280 - 1225 * c) + abs(297 - 1764 * c)
  return(f.c)

}
```

```
list.endpoints <- matrix(c(0, 1764), nrow = 1, byrow=T)

for (i in 1:iter) {
  new.endpoints <- golden.endpoints(list.endpoints[i,])
  new.endpoints <- test.endpoints(new.endpoints, list.endpoints[i,])
  list.endpoints <- rbind(list.endpoints, new.endpoints)
}
list.endpoints
```

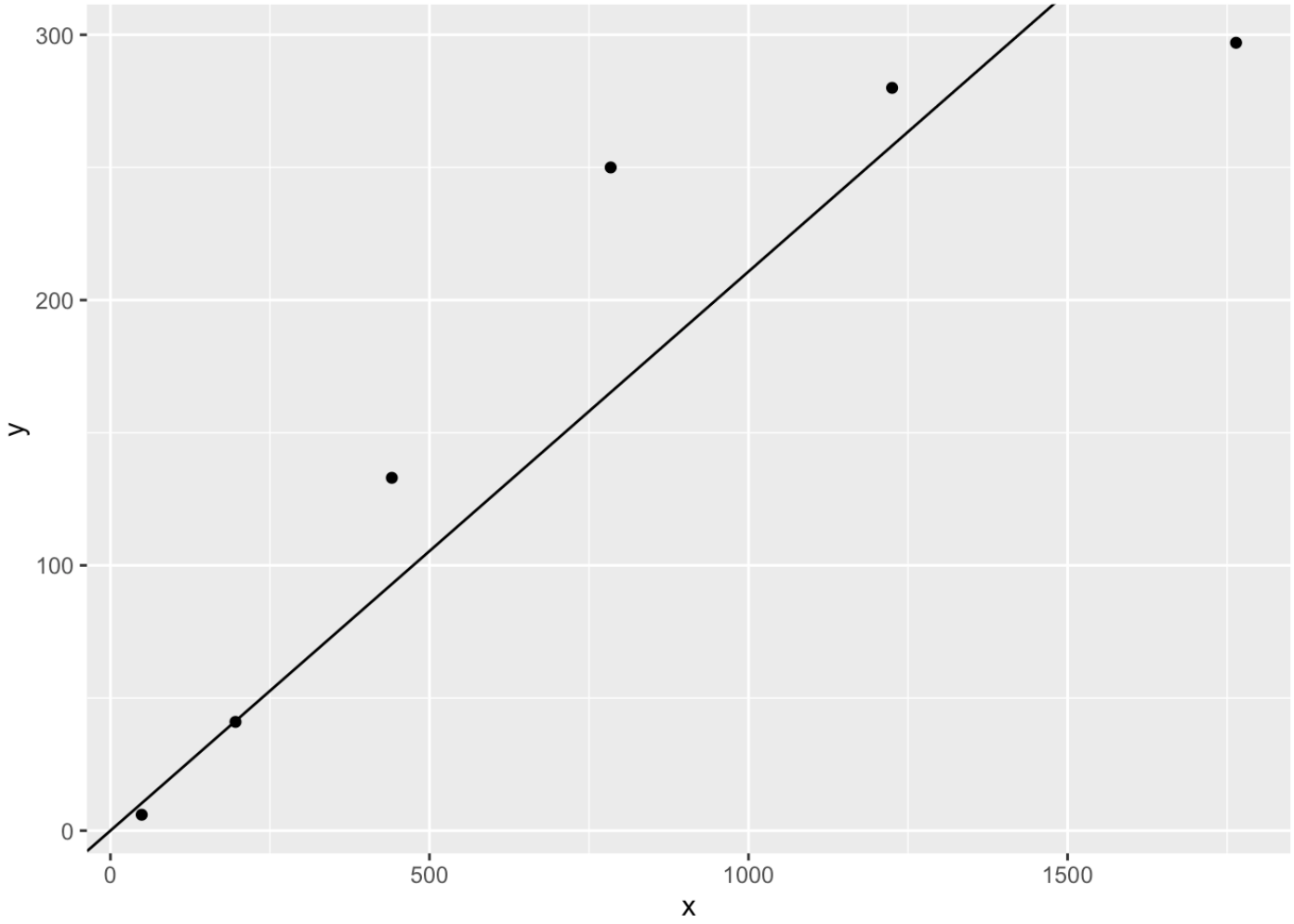
```
##            [,1]         [,2]
##  [1,] 0.0000000 1764.0000000
##  [2,] 0.0000000 1090.1520000
##  [3,] 0.0000000  673.7139360
##  [4,] 0.0000000  416.3552124
##  [5,] 0.0000000  257.3075213
##  [6,] 0.0000000  159.0160482
##  [7,] 0.0000000   98.2719178
##  [8,] 0.0000000   60.7320452
##  [9,] 0.0000000   37.5324039
## [10,] 0.0000000   23.1950256
## [11,] 0.0000000   14.3345258
## [12,] 0.0000000    8.8587370
## [13,] 0.0000000    5.4746994
## [14,] 0.0000000    3.3833643
## [15,] 0.0000000    2.0909191
## [16,] 0.0000000    1.2921880
## [17,] 0.0000000    0.7985722
## [18,] 0.0000000    0.4935176
## [19,] 0.0000000    0.3049939
## [20,] 0.1165077    0.3049939
```

```
slope <- mean(list.endpoints[iter+1,])
slope
```

```
## [1] 0.2107508
```

```
data <- as.data.frame(cbind(x, y))

ggplot(data, aes(x=x, y=y)) + geom_point() + geom_abline(slope = slope, intercept
= 0)
```



c

$$y = ax^3$$

```
x <- c(7, 14, 21, 28, 35, 42) ** 3


t = 0.2

iterations.estimate <- log(t / (74088)) / log(0.618)
iter <- round(iterations.estimate, 0)
iter
```

```
## [1] 27
```

```
error.model <- function(c){

  f.c <- abs(6 - 343 * c) + abs(41 - 2744  * c) + abs(133 - 9261 * c) + abs(250 -
21952 * c) + abs(280 - 42875 * c) + abs(297 - 74088 * c)
  return(f.c)

}
```

```
list.endpoints <- matrix(c(0, 74088), nrow = 1, byrow=T)

for (i in 1:iter) {
  new.endpoints <- golden.endpoints(list.endpoints[i,])
  new.endpoints <- test.endpoints(new.endpoints, list.endpoints[i,])
  list.endpoints <- rbind(list.endpoints, new.endpoints)
}
list.endpoints
```

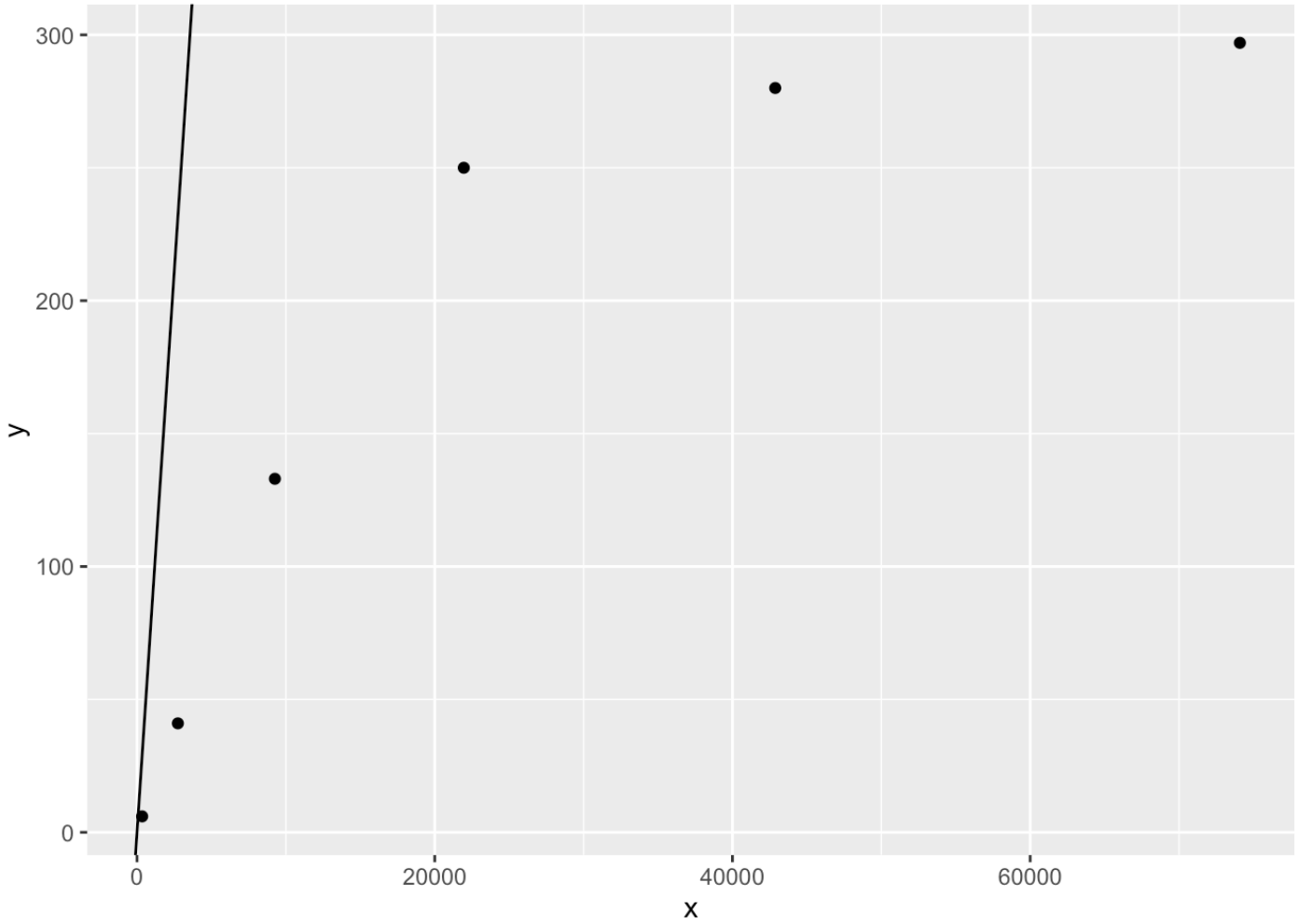
```
##       [,1]         [,2]
##  [1,]    0 7.408800e+04
##  [2,]    0 4.578638e+04
##  [3,]    0 2.829599e+04
##  [4,]    0 1.748692e+04
##  [5,]    0 1.080692e+04
##  [6,]    0 6.678674e+03
##  [7,]    0 4.127421e+03
##  [8,]    0 2.550746e+03
##  [9,]    0 1.576361e+03
## [10,]    0 9.741911e+02
## [11,]    0 6.020501e+02
## [12,]    0 3.720670e+02
## [13,]    0 2.299374e+02
## [14,]    0 1.421013e+02
## [15,]    0 8.781860e+01
## [16,]    0 5.427190e+01
## [17,]    0 3.354003e+01
## [18,]    0 2.072774e+01
## [19,]    0 1.280974e+01
## [20,]    0 7.916421e+00
## [21,]    0 4.892348e+00
## [22,]    0 3.023471e+00
## [23,]    0 1.868505e+00
## [24,]    0 1.154736e+00
## [25,]    0 7.136270e-01
## [26,]    0 4.410215e-01
## [27,]    0 2.725513e-01
## [28,]    0 1.684367e-01
```

```
slope <- mean(list.endpoints[iter+1,])
slope
```

```
## [1] 0.08421835
```

```
data <- as.data.frame(cbind(x, y))

ggplot(data, aes(x=x, y=y)) + geom_point() + geom_abline(slope = slope, intercept
= 0)
```



I'm not sure if this is a bug in my code or a really bad model.