Math 365 / Comp 365: Homework 6

Lawson Busch

Worked with Raven McKnight

Please bring a stapled hard copy of your answers to class

The following line sources functions from the class file `365Functions.r`. Feel free to use any of these functions throughout the semester.

```
source("https://drive.google.com/uc?export=download&id=10dNH3VbvxS8Z3OHjP4i9gRbtsf91VVBb")
require(Matrix)
```

Problem 1

A square matrix P is a ***projector*** if $P^2 = P$. A projector is an ***orthogonal projector*** if $P^T = P$. Let A be an $m \times n$ matrix with $m \geq n$. Recall from class that we showed that the orthogonal projection of a vector b onto the column space of A is given by

$$\hat{b} = AA^+b = A(A^TA)^{-1}A^Tb = Pb,$$

where A^+ is the pseudoinverse of A. Show that according to the definitions above, the matrix $P = A(A^TA)^{-1}A^T$ is indeed an orthogonal projector.

Hint: the transpose of the inverse of an invertible matrix is the inverse of the transpose of that invertible matrix.

To show that the matrix $P = A(A^TA)^{-1}A^T$ is a projector we must prove that $P^2 = P$:

$$P^2 = A(A^TA)^{-1}A^TA(A^TA)^{-1}A^T$$

And we know that:

$$(A^TA)^{-1}A^TA = I$$

So:

$$P^2 = AI(A^TA)^{-1}A^T = A(A^TA)^{-1}A^T = P$$

Which proves that $P^2 = P$.

And to prove that P is an orthogonal projector we must prove that $P^T = P$.

$$P^T = (A(ATA)^{-1}A^T)^T$$

If we let $C = A(ATA)^{-1}$ then:

$$P^T = AC^T$$

And if $B = ((A^TA)^{-1})^T$ then:

$$C^T = B^TA^T$$

Which means:

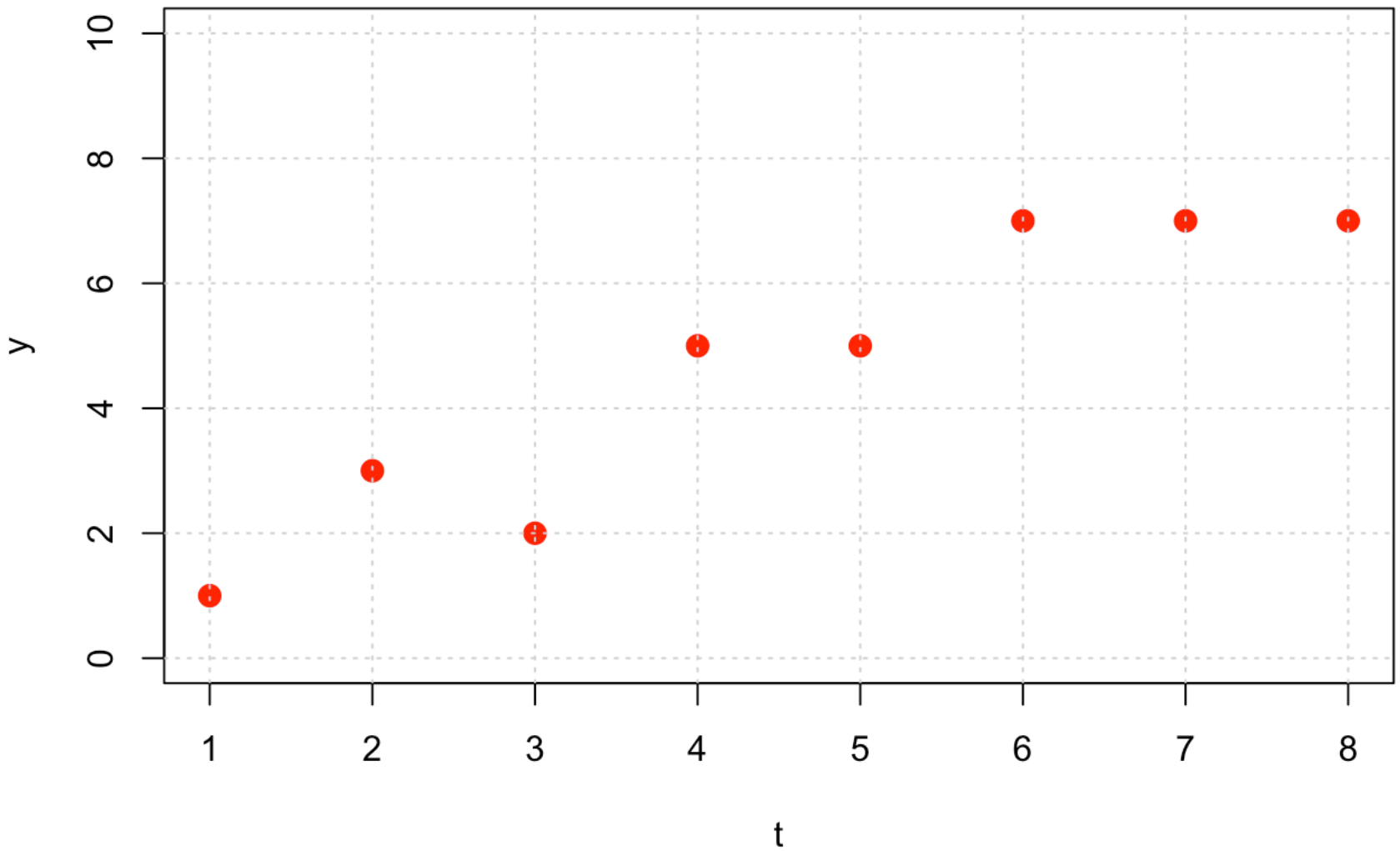
$$P^T = AB^TA^T = A(A^TA)^{-1}A^T = P$$

Problem 2

Note: This is Exercise 2 from Activity 17

Consider the points:

```
t=1:8
y=c(1,3,2,5,5,7,7,7)
plot(t,y,col='red',pch=20,cex=2,xlim=c(1,8),ylim=c(0,10))
grid()
```



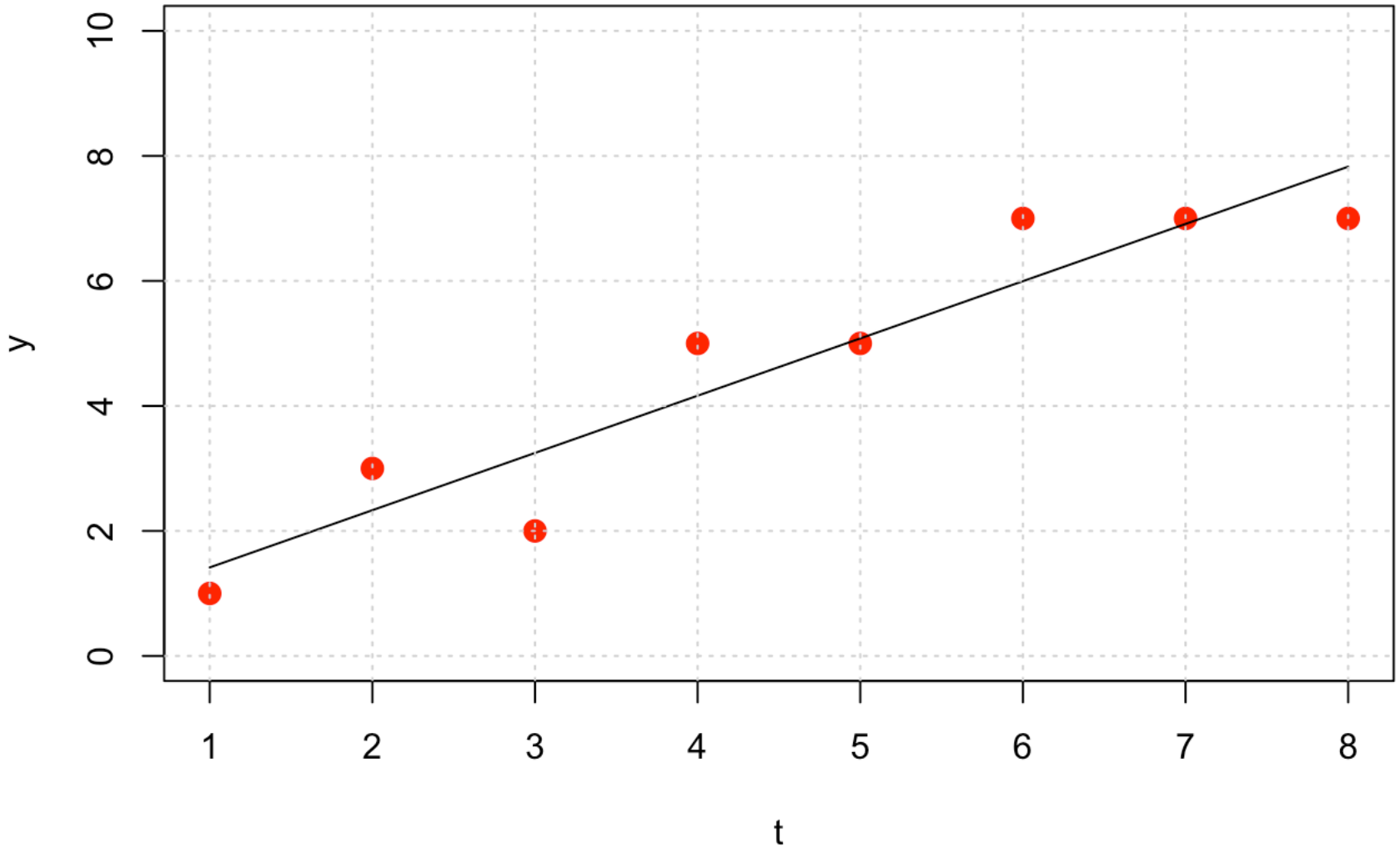
a. Setup and solve the normal equations in order to find the least squares line fit for this data. Plot your regression line on the same plot with the points.

```
ones = rep(1, 8)
A1 = cbind(ones, t)
b1 = y
x1 = solve(t(A1)%*%A1, t(A1)%*%b1)
x1
```

```
##           [,1]
## ones 0.5000000
## t    0.9166667
```

```
lsLine = function(x){0.916*x+0.5}
xx = seq(1, 8, length=10)

t=1:8
y=c(1,3,2,5,5,7,7,7)
plot(t,y,col='red',pch=20,cex=2,xlim=c(1,8),ylim=c(0,10))
lines(xx, lsLine(xx))
grid()
```



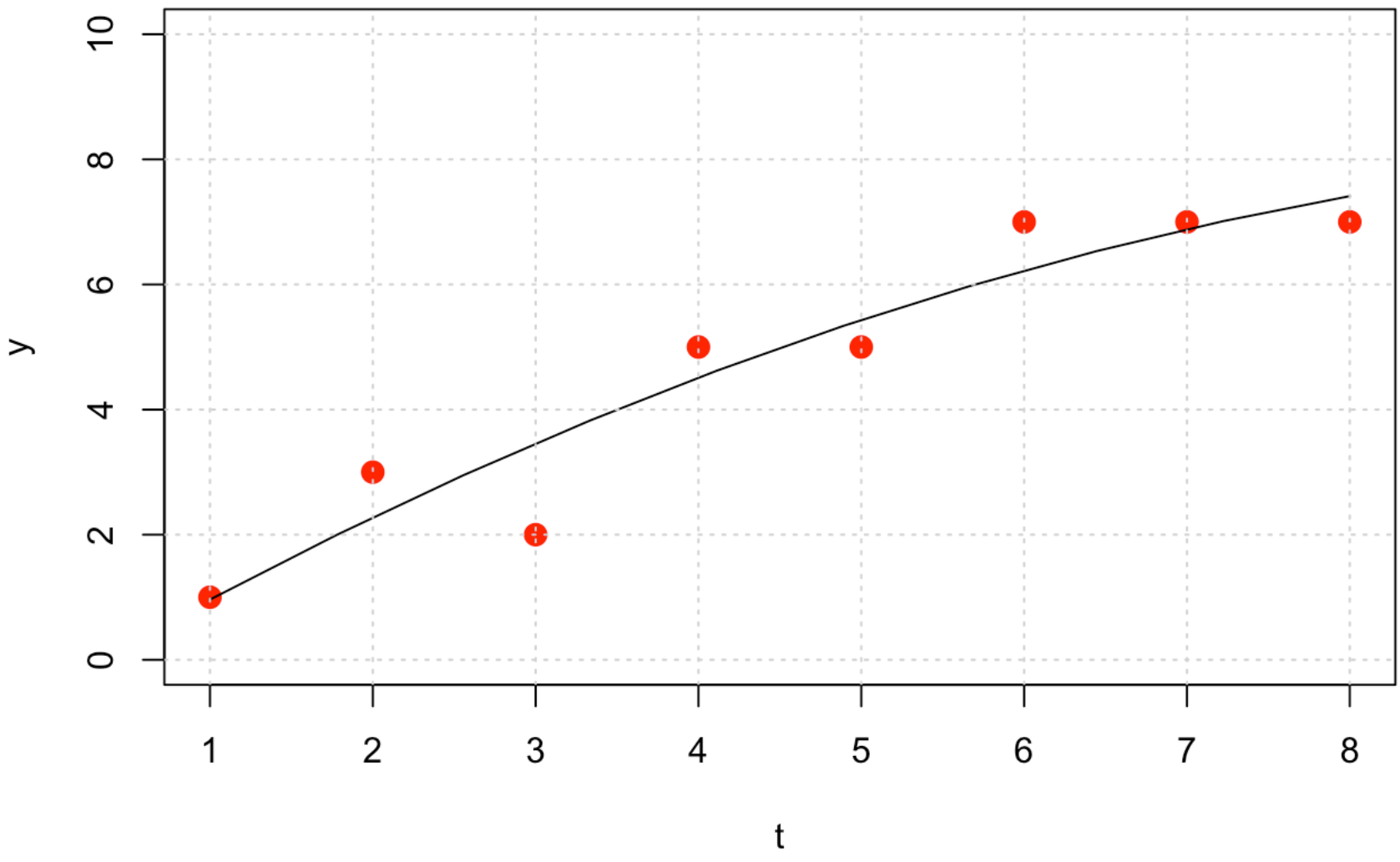
b. Setup and solve the normal equations in order to find the least squares parabola fit (i.e., polynomial of degree 2) for this data. Plot your best fit curve on the same plot with the points.

```
ones = rep(1, 8)
A2 = cbind(ones, t, t^2)
b2 = y
x2 = solve(t(A2)%*%A2, t(A2)%*%b2)
x2
```

```
##              [,1]
## ones -0.48214286
## t     1.50595238
##      -0.06547619
```

```
lsParab = function(x){-0.0654*x^2+1.51*x-0.482}
xx = seq(1, 8, length=10)

t=1:8
y=c(1,3,2,5,5,7,7,7)
plot(t,y,col='red',pch=20,cex=2,xlim=c(1,8),ylim=c(0,10))
lines(xx, lsParab(xx))
grid()
```



c. Do you expect $||Ax_* - b||_2^2$ (the sum of squared residuals) to be lower for part a) or part b)? Why?

Explain in plain English and using linear algebra vocabulary. Does your answer depend on the set of data given to you? Compute the sum of squared residuals for part a) and part b) to see if it matches your intuition.

I would expect $||Ax_* - b||_2^2$ to be lower for part b than part a of this question because part b adds another variable for us to use in our least squares approximation for x^* . I would expect this addition to lead to a more accurate prediction since $n <= m$, meaning that there are no free variables and thus the parabolic model should be more accurate. I do not think that the data would affect this, as even if the data was in a strait line, the degree two appoximation could simply solve it as a strait line (with the coefficient of x^2 being a vector of 0), making it as good as the degree one approximation.

```
r1 = A1%*%x1 - b1
r2 = A2%*%x2 - b2
res1 = t(r1) %*% r1
res2 = t(r2) %*% r2
res1 #Residual errors for degree one approximation
```

```
##          [,1]
## [1,] 4.583333
```

```
res2 #Residual errors for degree two approximation
```

```
##          [,1]
## [1,] 3.863095
```

After calculating the sum of squared residuals, I can see that my intuition was correct and that the sum of squared residuals is lower for the degree two approximation than the degree one approximation.

Problem 3

Note: This is Exercise 3 from Activity 17

In this exercise, we use optimization to show that a vector x that satisfies the normal equations represents the coefficients of the least squares solution.

a. Show that the least squares criterion $\frac{1}{2}||Ax - b||_2^2$ can be written as a quadratic function

$$f(x) = \frac{1}{2}x^\top Px + q^\top x + c.$$

That is, find the matrix P, vector q and constant c in terms of A and b. Hint: remember that for any vector v, $||v||_2^2 = v^\top v$.

$$\frac{1}{2}||Ax - b||_2^2 = \frac{1}{2}((Ax - b)^T(Ax - b))$$

$$\frac{1}{2}(((Ax)^T - b^T)(Ax - b))$$

$$\frac{1}{2}((x^TA^T - b^T)(Ax - b))$$

$$\frac{1}{2}(x^TA^T(Ax - b) - b^T(Ax - b))$$

$$\frac{1}{2}(x^TA^TAx - x^TA^Tb - b^TAx + b^Tb)$$

$$\frac{1}{2}(x^TA^TAx - (b^TAx)^T - b^TAx + b^Tb)$$

$$\frac{1}{2}(x^TA^TAx - 2b^TAx + b^Tb)$$

$$\frac{1}{2}x^TA^TAx - b^TAx + \frac{1}{2}b^Tb$$

Which implies that $P = A^TA$, $q^T = -b^TA$, and c=$\frac{1}{2}b^Tb$.

b. Show that the P matrix from part (a) is symmetric and positive semidefinite.

We know that P is symmetric because $P = A^TA$ and $(A^TA)^T = A^TA$, implying that it is symmetric. Further, we know that P is positive semidefinite because:

$$x^T(A^TA)x = (x^TA^T)Ax = (Ax)^TAx = ||Ax||_2^2$$

And we know that:

$$||Ax||_2^2 >= 0$$

Proving that P is symmetric positive semidefinite.

c. Recall that if the P matrix in the quadratic function form above is symmetric positive semidefinite, then $\nabla f(x) = Px + q$, and the critical point x satisfying $\nabla f(x) = 0$ is a global minimum. Conclude that the value of x that satisfies the normal equations gives the optimal least squares coefficients.

The value of x that satisfies the normal equations will produce the optimal least squares coefficients because the normal equations produce the x with the minimum residual. This implies that the coefficients produced for this x will be optimal, as the x that satisfies the normal equations will also be the x that minimizes the gradient and sets the global minimum.

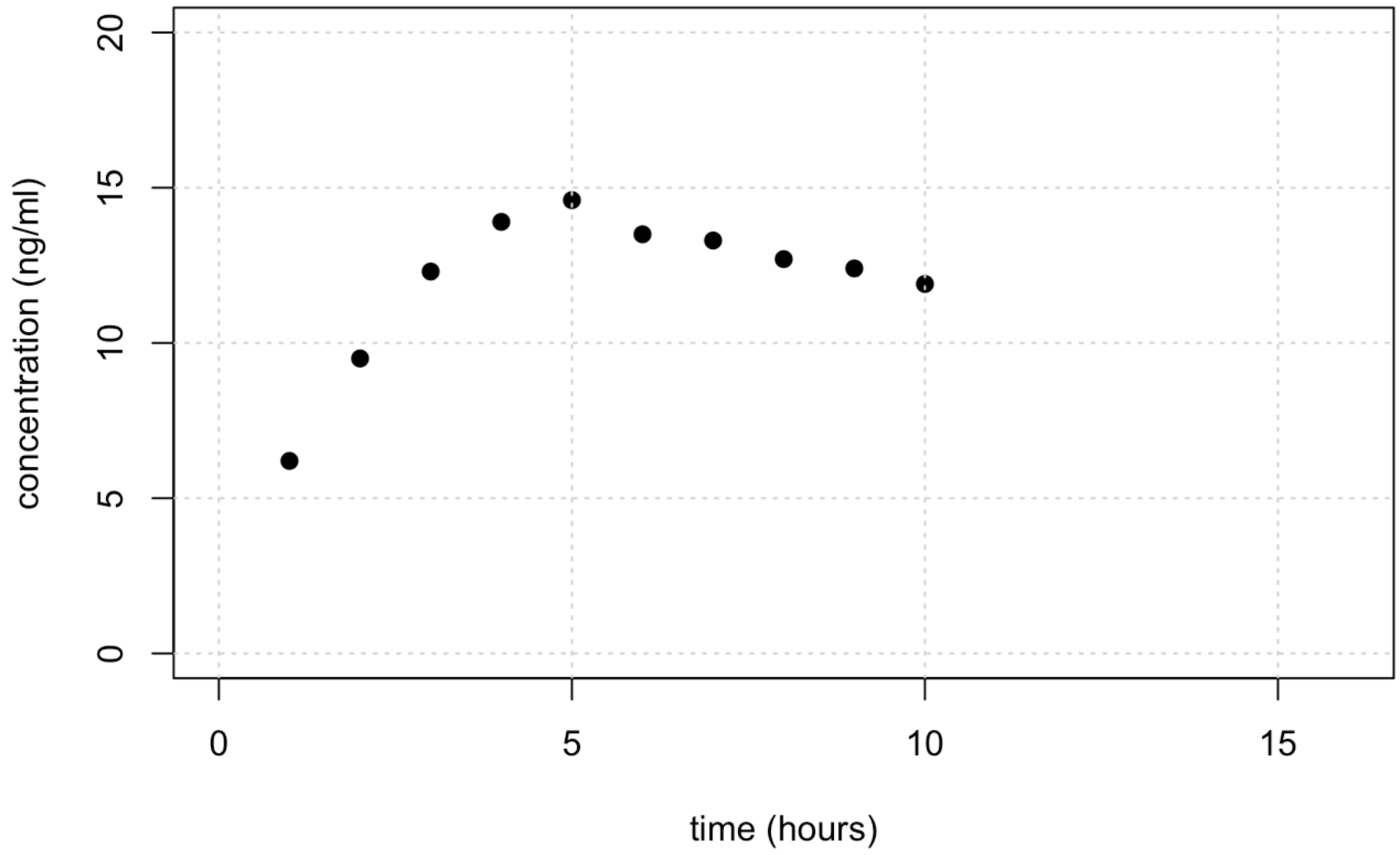
Problem 4

Note: Before doing this problem, read Section 4.2.2 in the book.

The data for a drug concentration model corresponding to Computer Problem 6 in Section 4.2 is here:

```
hour = (1:10)
concentration = c(6.2,9.5,12.3,13.9,14.6,13.5,13.3,12.7,12.4,11.9)

plot(hour,concentration,pch=19,col='black',
     xlim=c(0,16),ylim=c(0,20),ylab="concentration (ng/ml)",xlab='time (hours)')
grid()
```



You'd like to fit a "surge" model of the form

$$y = c \cdot t \cdot e^{k \cdot t}$$

The problem is that this is not linear.

But logarithms come to the rescue!

$$\ln(y) = \ln(c \cdot t \cdot e^{k \cdot t}) = \ln(c) + \ln(t) + k \cdot t,$$

so

$$\ln(c) + k \cdot t = \ln(y) - \ln(t)$$

The LHS is linear in t. The RHS is a constant (since both y and t are given). The unknowns are $\ln(c)$ and k.

Use a least-squares fit of a line to this data to get $\ln(c)$ and k. Then reassemble the answer into the function

$y = c \cdot t \cdot e^{k \cdot t}$ and plot it along with the data.

Report your values of c and k and give a plot.

```
A = cbind(c(1), hour)
b = log(concentration) - log(hour)
solve(t(A)%*%A, t(A)%*%b)
```

```
##            [,1]
##       1.963192
## hour -0.183849
```

We know that ln(c) = 1.96319 and that k = -0.18384. So if we remove the ln by raising to the power of c, we know that c = 7.122.

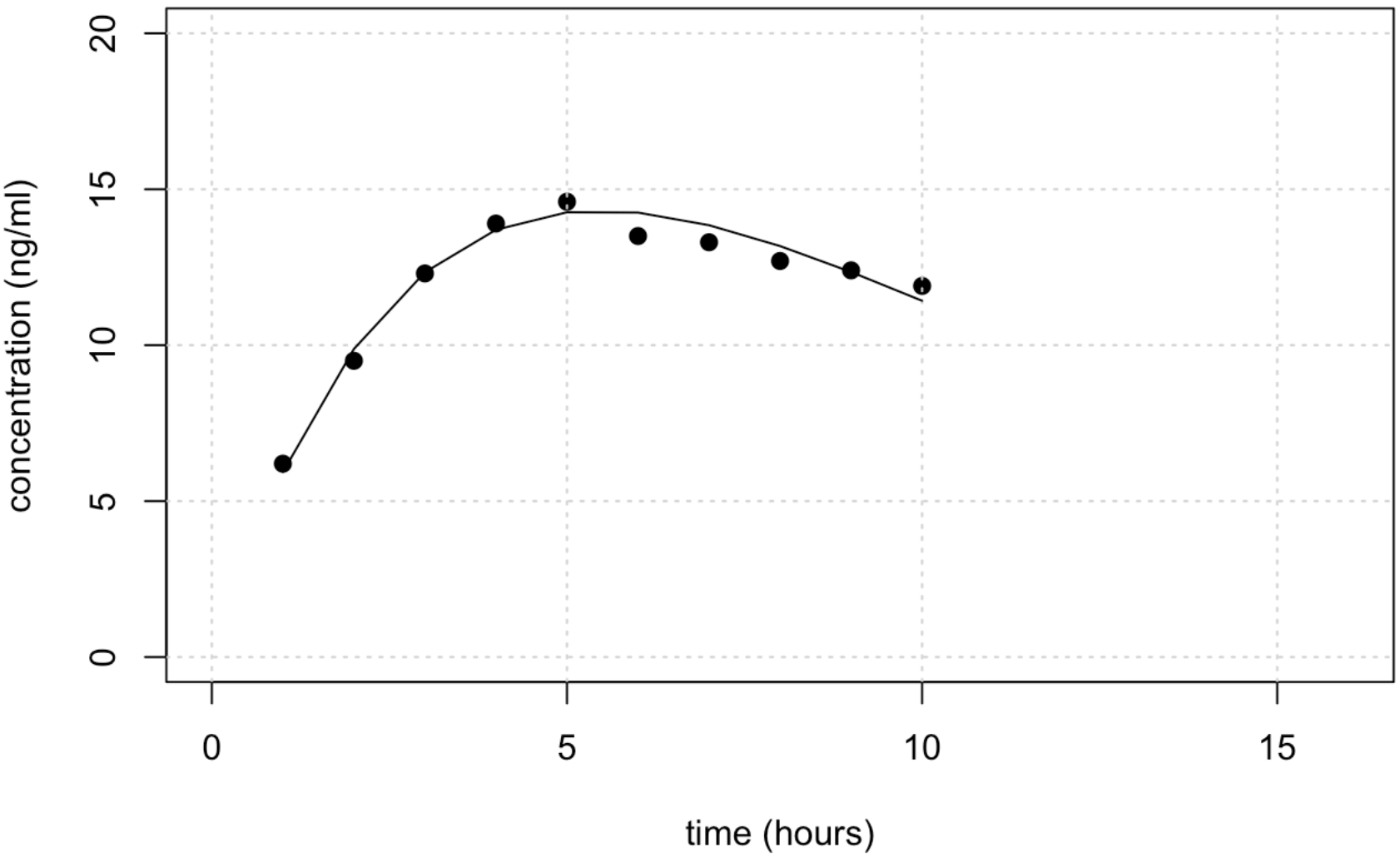
So if we plug these values back into our original function we get:

$$y = 7.122 \cdot t \cdot e^{-0.183 \cdot t}$$

So to plot the function:

```
y = function(t) {
  val = 7.122*t*exp(-0.183*t)
  return(val)
}

plot(hour,concentration,pch=19,col='black',
     xlim=c(0,16),ylim=c(0,20),ylab="concentration (ng/ml)",xlab='time (hours)')
grid()
lines(y(hour))
```



Problem 5

Note: This is Exercise 1 from Activity 18

You should do part (a) of this problem ***by hand***.

a. Use the classical Gram-Schmidt orthogonalization algorithm to find the reduced QR factorization and full QR factorization of the matrix

$$A = \begin{pmatrix} 2 & 3 \\ -2 & -6 \\ 1 & 0 \end{pmatrix}.$$

To find the reduced QR factorization we must solve:

$$q_1 = \frac{1}{||y_1||_2} y_1, \; where \; y_1 = v_1$$

$$q_1 = \frac{1}{\sqrt{2 * 2 + (-2) * (-2) + 1 * 1}} \begin{pmatrix} 2 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ \frac{-2}{3} \\ \frac{1}{3} \end{pmatrix}$$

Then we must solve for y_2 where:

$$y_2 = v_2 - q_0 * q_0^T * v_2$$

$$y_2 = \begin{pmatrix} 3 \\ -6 \\ 0 \end{pmatrix} - \begin{pmatrix} \frac{2}{3} \\ \frac{-2}{3} \\ \frac{1}{3} \end{pmatrix} * \begin{pmatrix} \frac{2}{3} & \frac{-2}{3} & \frac{1}{3} \end{pmatrix} * \begin{pmatrix} 3 \\ -6 \\ 0 \end{pmatrix}$$

$$y_2 = \begin{pmatrix} 3 \\ -6 \\ 0 \end{pmatrix} - \begin{pmatrix} \frac{4}{9} & -\frac{4}{9} & \frac{2}{9} \\ -\frac{4}{9} & \frac{4}{9} & -\frac{2}{9} \\ \frac{2}{9} & -\frac{2}{9} & \frac{1}{9} \end{pmatrix} * \begin{pmatrix} 3 \\ -6 \\ 0 \end{pmatrix}$$

$$y_2 = \begin{pmatrix} 3 \\ -6 \\ 0 \end{pmatrix} - \begin{pmatrix} 4 \\ -4 \\ 2 \end{pmatrix} = \begin{pmatrix} -1 \\ -2 \\ -2 \end{pmatrix}$$

And we know that:

$$q_2 = \frac{1}{||y_2||_2} y_2$$

$$q_2 = \frac{1}{\sqrt{(-1) * (-1) + (-2) * (-2) + (-2) * (-2)}} \begin{pmatrix} -1 \\ -2 \\ -2 \end{pmatrix} = \begin{pmatrix} -\frac{1}{3} \\ -\frac{2}{3} \\ -\frac{2}{3} \end{pmatrix}$$

So we know that the reduced QR factorization is:

$$q_1 = \begin{pmatrix} \frac{2}{3} \\ \frac{-2}{3} \\ \frac{1}{3} \end{pmatrix}, q_2 = \begin{pmatrix} -\frac{1}{3} \\ -\frac{2}{3} \\ -\frac{2}{3} \end{pmatrix}$$

To find the full QR facorization we must find a basis for $null(A^T)$. We can do this by putt A^T in $RREF$ and solving the parametric vector form. Where:

$$A^T = \begin{pmatrix} 2 & -2 & 1 \\ 3 & -6 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1/2 \end{pmatrix}$$

We then know that $x_1 = -x_3$, $x_2 = -\frac{1}{2}$, $x_3 = x_3$ from the parametric vector equations. As a result, we know that the final column in our full QR factorization is this x_3. Or, that the last column is equal to:

$$q_3 = \begin{pmatrix} -1 \\ -\frac{1}{2} \\ 1 \end{pmatrix}$$

Which is the final column of our full QR factorization.

b. First check your answers by checking that $A = QR$ and $A = \bar{Q}\bar{R}$. Then check if you computed the same factorizations as the `qr` function in R, which you can do with the following code

```
A = cbind(c(2, -2, 1), c(3, -6, 0))
A
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]   -2   -6
## [3,]    1    0
```

```
# Reduced QR
out=qr(A)
(R=qr.R(out))
```

```
##      [,1] [,2]
## [1,]   -3   -6
## [2,]    0    3
```

```
(Q=qr.Q(out))
```

```
##            [,1]       [,2]
## [1,] -0.6666667 -0.3333333
## [2,]  0.6666667 -0.6666667
## [3,] -0.3333333 -0.6666667
```

```
Q%*%R
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]   -2   -6
## [3,]    1    0
```

Note that R's `qr` algorithm does not ensure that all of the diagonal entries of R are nonnegative (in which case the factorization is not unique). If you want to force the R matrix to have positive diagonals, you can form a diagonal matrix S whose i^{th} diagonal is equal to the sign of the i^{th} diagonal of R. Then let $\tilde{Q} = QS$ and $\tilde{R} = SR$, so that $\tilde{Q}\tilde{R} = QS^2R = QR = A$ (since $S^2 = I$).

```
# Fix signs
s=sign(diag(R))
S=diag(s)
Q.tilde=Q%*%S
R.tilde=S%*%R
Q.tilde%*%R.tilde
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]   -2   -6
## [3,]    1    0
```

```
# Full QR
(R.bar=qr.R(out,complete=TRUE))
```

```
##      [,1] [,2]
## [1,]   -3   -6
## [2,]    0    3
## [3,]    0    0
```

```
(Q.bar=qr.Q(out,complete=TRUE))
```

```
##            [,1]       [,2]       [,3]
## [1,] -0.6666667 -0.3333333 -0.6666667
## [2,]  0.6666667 -0.6666667 -0.3333333
## [3,] -0.3333333 -0.6666667  0.6666667
```

```
Q.bar%*%R.bar
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]   -2   -6
## [3,]    1    0
```

c. Use the reduced QR factorization of A from part (a) to find the least squares solution to

$$\begin{pmatrix} 2 & 3 \\ -2 & -6 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3 \\ -3 \\ 6 \end{pmatrix}.$$

```
b = c(3, -3, 6)
outC = qr(A)
solve(qr.R(outC), t(qr.Q(outC))%*%b)
```

```
##      [,1]
## [1,]    4
## [2,]   -1
```

```
#qr.solve(outC, b)
```

d. You can check that $\bar{Q}^\top b = \begin{bmatrix} 6 \\ -3 \\ -3 \end{bmatrix}$. Without doing extra computations (i.e., do not actually multiply out $Ax_* - b$), what is the squared error $||Ax_* - b||_2^2$ associated with the least squares solution in part (c)?

So the least squares error assosiacted with $\hat{Q}$ is 9, since the portion of $\bar{Q}$ that is $\hat{Q}$ is simply the last column in the result of $\bar{Q} * b$, and we know that the least square error is $||\hat{Q}^T * b||^2$, so since the norm of the last column is 3, we know the least squares error is 9.

e. Recall that if A is an $m \times n$ matrix, the null space of $A^\top$ is the orthogonal complement of the column space of A and $\dim(\text{null}(A^\top)) + \dim(\text{col}(A)) = m$. Use the full QR factorization of the A matrix above to find a basis for the null space of $A^\top$.

```
Q.bar[,3]
```

```
## [1] -0.6666667 -0.3333333  0.6666667
```

We know that the orthogonal complement of the column space in the QR matrix is the set of columns that make up $\hat{Q}$. In this case we know that $\hat{Q}$ is only the last column of $\bar{Q}$. Thus by pulling the last column of $\bar{Q}$, we can find a basis for the nullspace of $A^\top$. This is the vector (-2/3, -1/3, 2/3).

Problem 6

Note: This is Exercise 3 from Activity 18

The vectors u and v below are the x-coordinates and y-coordinates of 50 points (u_i, v_i) in the plane. We want to fit a circle to these points. Denote the center of the circle by (u_c, v_c) and the radius by R. A point (u, v) is on the circle if $(u - u_c)^2 + (v - v_c)^2 = R$. We can therefore formulate the fitting problem as

$$\min_{u_c,\, v_c,\, R} \left\{ \sum_{i=1}^{50} \left[(u_i - u_c)^2 + (v_i - v_c)^2 - R^2 \right]^2 \right\}.$$

If we do a change of variable $w = u_c^2 + v_c^2 - R^2$, then we can write the above problem as a linear least squares problem

$$\min_x ||Ax - b||^2,$$

where $x = \begin{bmatrix} u_c \\ v_c \\ w \end{bmatrix}$.

```
u = c(-3.9265307,-3.1716160e+00,-1.6115988e+00,-2.6679398e+00,-1.7299714e+00,-2.21850
18e+00,-2.0618500e+00,-1.4774499e+00,-3.2095408e+00,-2.0139385e+00,-2.0965393e+00,-2.
8414848e+00,-3.5516322e+00,-2.3325005e+00,-1.6889345e+00,-1.4937155e+00,-1.3103945e+0
0,-1.3082423e+00,-1.5221371e+00,-1.8621796e+00,-2.8784185e+00,-3.3058351e+00,-2.94181
36e+00,-3.5689305e+00,-3.2715656e+00,-1.8167830e+00,-2.6160985e+00,-3.6369299e+00,-3.
6094960e+00,-3.8213899e+00,-3.5639197e+00,-2.9667150e+00,-1.9473222e+00,-3.0470691e+0
0,-2.8955875e+00,-3.2029692e+00,-2.2688964e+00,-2.3212990e+00,-1.1585153e+00,-1.89934
55e+00,-3.5771792e+00,-2.6473229e+00,-1.4699478e+00,-3.7978927e+00,-2.0968345e+00,-4.
0118440e+00,-2.2415905e+00,-1.3737454e+00,-2.0935937e+00,-1.4260492e+00)

v = c(5.7992251e+00,7.3130620e+00,7.5592434e+00,7.6911348e+00,5.5113079e+00,7.7442101
e+00,7.7091849e+00,6.0549104e+00,7.5170875e+00,7.6045473e+00,5.1354212e+00,
5.0671844e+00,7.3910732e+00,7.6949226e+00,5.3469286e+00,7.3473664e+00,6.8715471e+00,6
.7842012e+00,5.7283630e+00,7.7633148e+00,7.7677261e+00,5.4778857e+00,5.0690285e+00,5.
5246190e+00,7.6772318e+00,5.3181407e+00,7.6148680e+00,7.3524730e+00,6.0303455e+00,5.8
476992e+00,5.8479253e+00,5.3237261e+00,5.1703804e+00,5.4245981e+00,7.7991795e+00,5.57
34007e+00,7.8705366e+00,5.1617927e+00,6.1579013e+00,5.4067639e+00,7.2445803e+00,7.680
5233e+00,6.1180277e+00,7.3691475e+00,7.6463880e+00,6.1479510e+00,7.7414349e+00,7.2054
473e+00,5.2385698e+00,5.8594283e+00)
```

a. Define A and b in this least squares formulation. What are their dimensions?

```
ones = rep(-1, 50) #need to figure out what to rep
A = cbind(2*u, 2*v, ones)
b = u^2+v^2
```

b. Solve the least squares problem to find u_c, v_c, and w (you can solve the normal equations or use `qr.solve`), and then use the change of variable equation to find R.

```
qr.solve(t(A)%*%A, t(A)%*%b)
```

```
##            [,1]
##      -2.567143
##       6.468049
## ones 46.679801
```

From our solution we can see that $u_c = -2.567$ and that $v_c = 6.468$. We also know that $w = 46.679$. So to solve for R we can rewrite $w = u_c^2 + v_c^2 - R^2$ as $R^2 = u_c^2 + v_c^2 - w$ and solve for R.

```
u1 = -2.5671433615624468239
v1 = 6.4680491161986637394
w = 46.6798009453701254756
r = sqrt(u1^2+v1^2-w)
r
```

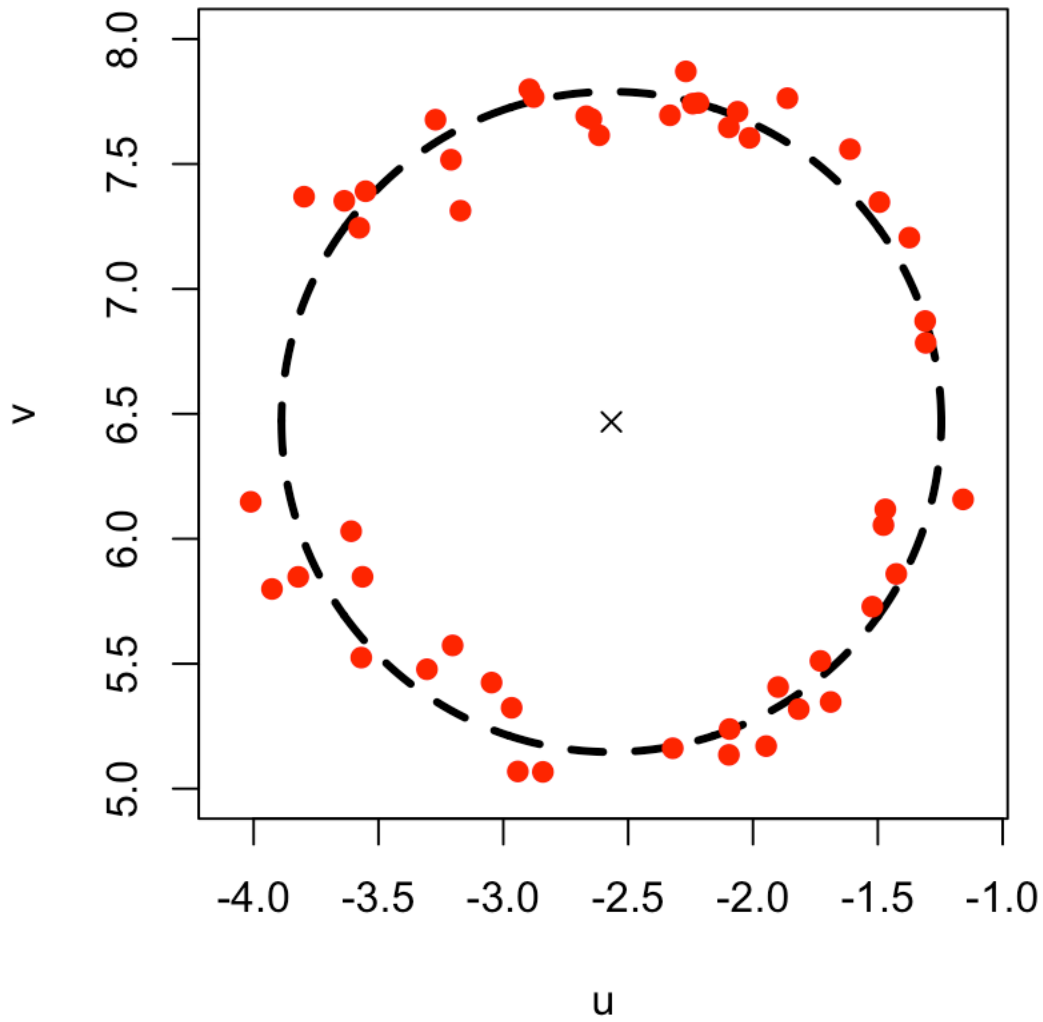
```
## [1] 1.321395
```

So we see that $R = 1.321$.

c. To double check your work, let's plot the data and the fitted circle using the following code:

```
# enter your values for uc, vc, and R here
uc=-2.5671433615624468239
vc=6.4680491161986637394
R=1.3213945145195391 54
```

```
# plot code
t=seq(0,2*pi,length=1000) # parameterize points on the circle
par(pty="s") # for the plot to be square
plot(R*cos(t)+uc,R*sin(t)+vc,type='l',lty=2,lwd=3,xlim=c(-4.1,-1.1),ylim=c(5,8),xlab=
"u",ylab="v") # plot the best fit line
points(u,v,pch=19,col="red") # plot the initial data points
points(uc,vc,pch=4) # plot an x at the center point (uc,vc)
```



Problem 7

Note: This is Part IV from Activity 19

Given some vector g in a vector space $\mathcal{V}$ and some subspace $\mathcal{S}$ of the vector space $\mathcal{V}$, the least squares problem is to find the vector $\hat{g} \in \mathcal{S}$ that is closest to g; that is,

$$\hat{g} = \text{argmin}_{f \in \mathcal{S}} \left\{ ||g - f||^2 \right\}.$$

In our previous work, the vector space $\mathcal{V}$ was $\mathbb{R}^m$ and we called the vector g by b; the norm used to measure "closest" was the Euclidean 2-norm; the subspace $\mathcal{S}$ was the column space of A for some matrix A; and we denoted the vector in $\mathcal{S}$ that is closest to b by $\hat{b}$.

In our new inner product space of continuous functions, $\mathcal{V} = C([a, b])$, $\mathcal{S}$ is some subspace of functions (often the space of polynomials of a given degree), and the squared error to be minimized is

$$||g - f||^2 = \langle g - f, g - f \rangle = \int_a^b [g(x) - f(x)]^2 \, dx.$$

The rest of the main ideas are exactly the same. The solution is still an orthogonal projection of g onto $\mathcal{S}$, which you know how to do if you have an orthonormal basis for $\mathcal{S}$. We found an orthogonal basis in Part I of the activity, which we learned in Part II is made up of elements called the Legendre polynomials.

a. Let $g(x) = x^2 + 2x - 3$ (the same quadratic function we used in Part IIIa of the activity). Find the least squares polynomial approximation of degree 1 (i.e., a line) to the function $g(x)$ on the interval $[-1, 1]$. That is, find

$$\hat{g} = \text{argmin}_{f \in \mathcal{P}_1} \left\{ ||g - f||^2 \right\} = \text{argmin}_{f \in \mathcal{P}_1} \left\{ \int_{-1}^{1} [g(x) - f(x)]^2 \right\}.$$

- Hint 1: The best fit line is NOT simply $2x - 3$, the lower order terms of $g(x)$. This is because the basis of monomials are NOT orthonormal!
- Hint 2: You should be able to find the answer without doing any additional computations beyond what you've already done for Part III.

```
q0 = function(x){1/sqrt(2)}
q1 = function(x){sqrt(3/2)*x}
g = function(x){x^2+2*x-3}

gq0 = function(x){1/sqrt(2)*x^2+2/sqrt(2)*x-3/sqrt(2)} #Inner product of q0 and g
gq1 = function(x){sqrt(3/2)*x^3+2*sqrt(3/2)*x^2-3*sqrt(3/2)*x} #Inner product of q1 a
nd g

a0 = integrate(gq0, -1, 1)
a1 = integrate(gq1, -1, 1)
a0
```

```
## -3.771236 with absolute error < 4.2e-14
```

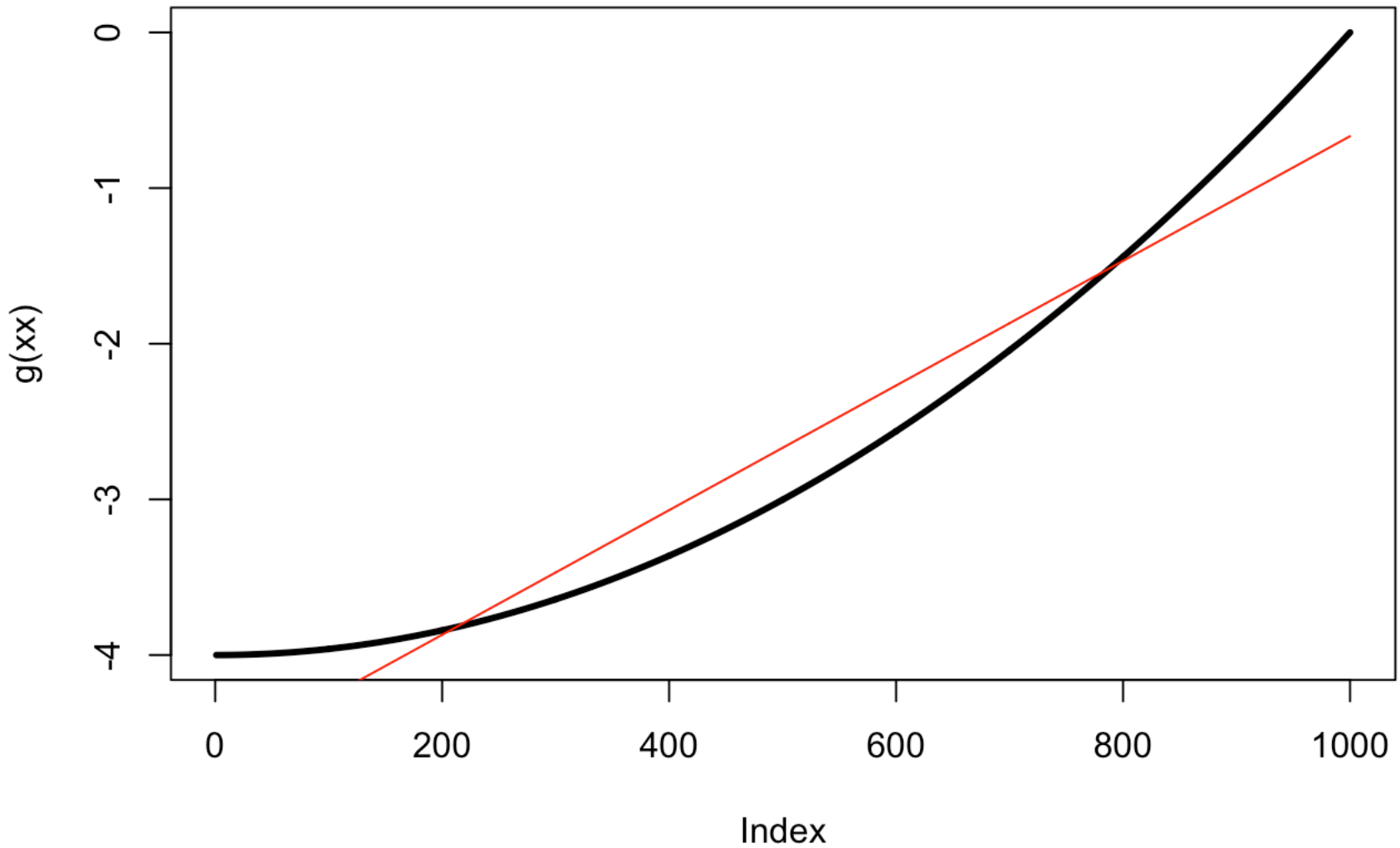
```
a1
```

```
## 1.632993 with absolute error < 3.4e-14
```

```
term1 = -3.7712361663282529811 * 1/sqrt(2)
term2 = 1.6329931618554518469 * sqrt(3/2)

fit = function(x){term1 + term2*x}

xx = seq(-1, 1, length=1000)
plot(g(xx),type='l',lwd=3)
lines(fit(xx), col='red')
```



To get the difference between the two functions $\hat{g}$ and g we:

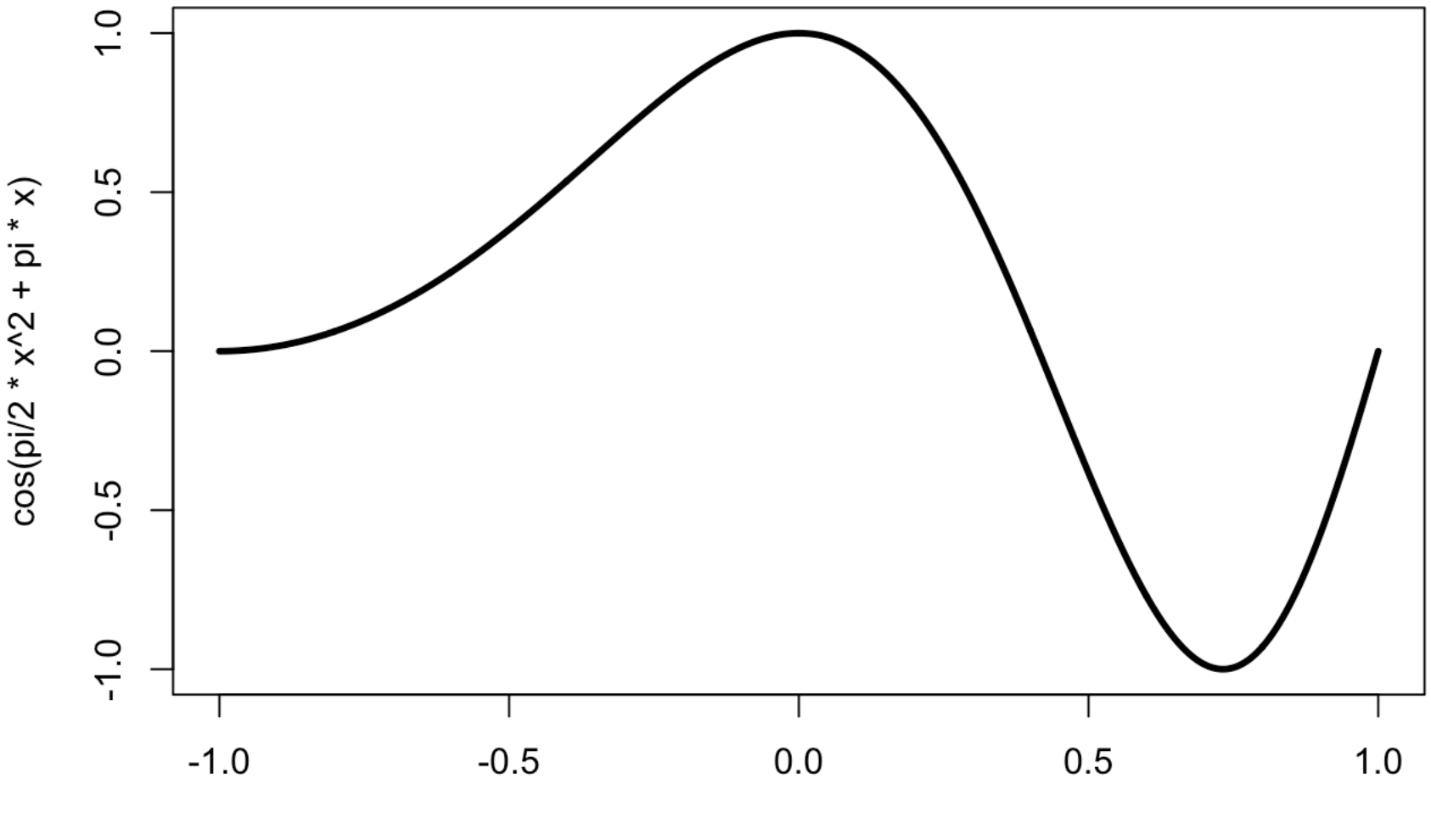
```
ghat_m_g = function(x){(term1 + term2*x - (x^2+2*x-3))^2}
integrate(ghat_m_g, -1, 1)
```

```
## 0.1777778 with absolute error < 2e-15
```

Or we get that the distance between our optimal first degree approximation of g, $\hat{g}$, and g is only 0.17777777777777781232.

b. Let $h(x) = \cos\left(\frac{\pi}{2}x^2 + \pi x\right)$. Here is a plot of $h(\cdot)$ on the interval $[-1, 1]$:

```
x=seq(-1,1,length=1000)
plot(x,cos(pi/2*x^2+pi*x),type='l',lwd=3)
```



Find

$$\hat{h} = \text{argmin}_{f \in \mathcal{P}_3} \left\{ \int_{-1}^{1} [h(x) - f(x)]^2 \right\},$$

and plot $\hat{h}(\cdot)$ on the same graph as $h(\cdot)$ on the interval $[-1, 1]$. How close (using the standard L^2 distance on functions) are $h(x)$ and $\hat{h}(x)$? Hint: feel free to use R or Wolfram Alpha to compute integrals.

To find the optimal polynomial that fits h we can again solve for a linear combination of the Legendre polynomials.

```
hq0 = function(x){1/sqrt(2)*cos(pi/2*x^2+pi*x)}
hq1 = function(x){sqrt(3/2)*x*cos(pi/2*x^2+pi*x)}
hq2 = function(x){sqrt(5/8)*(3*x^2-1)*cos(pi/2*x^2+pi*x)}
hq3 = function(x){sqrt(7/8)*(5*x^3-3*x)*cos(pi/2*x^2+pi*x)}

a0 = integrate(hq0, -1, 1)
a1 = integrate(hq1, -1, 1)
a2 = integrate(hq2, -1, 1)
a3 = integrate(hq3, -1, 1)

a0
```

```
## 0.2428316 with absolute error < 1.2e-09
```

```
a1
```

```
## -0.4205966 with absolute error < 1.8e-09
```

```
a2
```

```
## -0.5982874 with absolute error < 4.8e-08
```

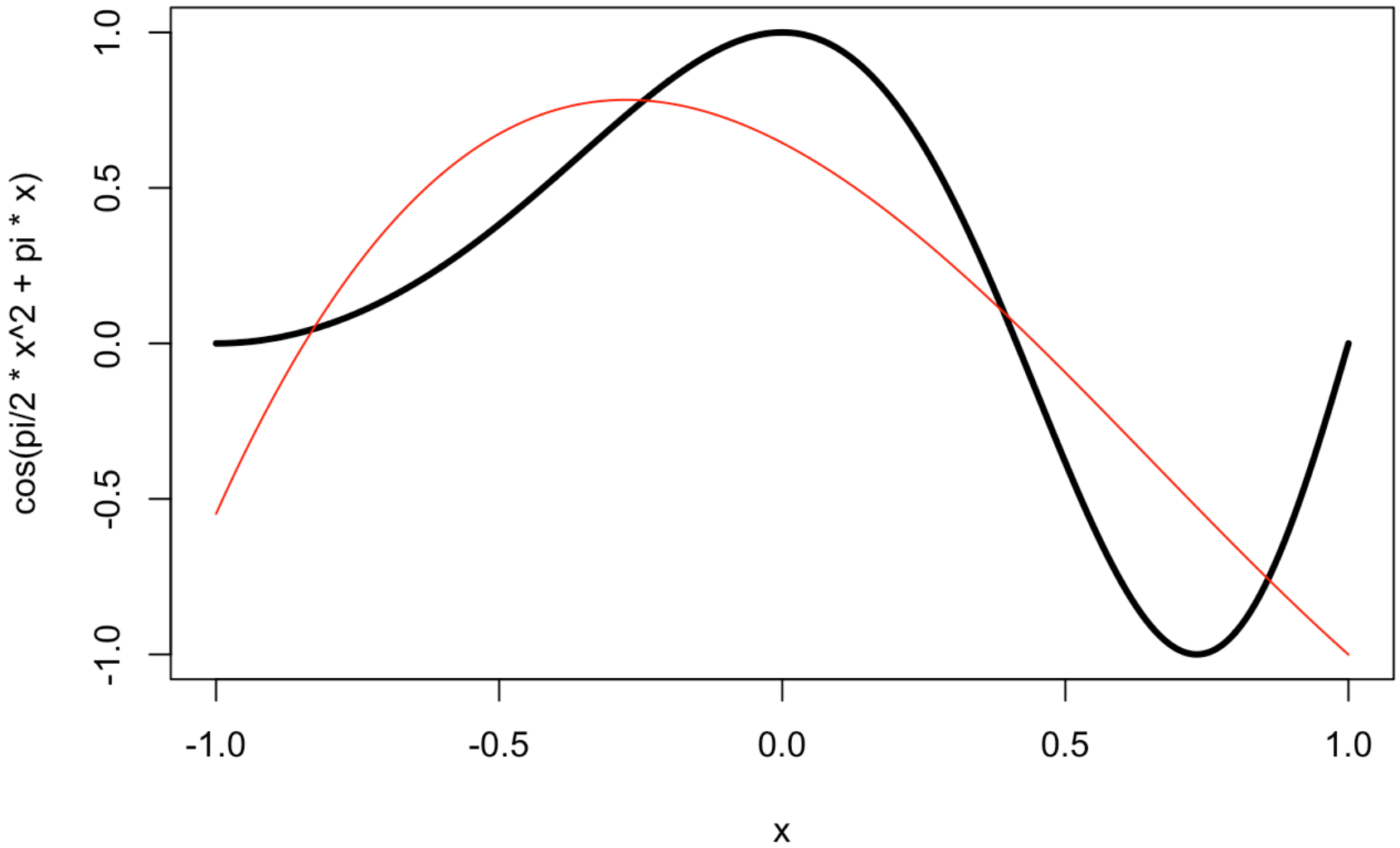
```
a3
```

```
## 0.1543707 with absolute error < 8.4e-07
```

```
c0 = 0.24283155493674946079 * 1/sqrt(2)
c1 = -0.42059659083140288338 * sqrt(3/2)
c2 = -0.59828736068750398847 * sqrt(5/8)
c3 = 0.15437069380063628277 * sqrt(7/8)

hfit = function(x){c0+c1*x+c2*(3*x^2-1)+c3*(5*x^3-3*x)}


x=seq(-1,1,length=1000)
plot(x,cos(pi/2*x^2+pi*x),type='l',lwd=3)
lines(x, hfit(x), col = "Red")
```



Then to find the difference between $\hat{h}$ and h:

```
hfit_m_h = function(x){(c0+c1*x+c2*(3*x^2-1)+c3*(5*x^3-3*x) - cos(pi/2*x^2+pi*x))^2}
integrate(hfit_m_h, -1, 1)
```

```
## 0.2071252 with absolute error < 1.2e-10
```

Or we get that the distance between the two functions is 0.20712524193855225141.