# Solution 1: Basic programming in R

Siyue Yang

06/11/2022

## Part 1: Matrix and vector operations.

1. Solve the following system:

$$\begin{bmatrix} a_1 & b_1 & & & 0 \\ c_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & a_{99} & b_{99} \\ 0 & & & c_{99} & a_{100} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{100} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{100} \end{bmatrix}$$

where

$$a_j = j, \quad b_j = 1/j, \quad c_j = 1, \quad d_j = \sin(j\pi/200)$$

and print $x_1, x_2, \ldots, x_5$.

**Solution**:

```
# Define A.
A <- matrix(rep(0, 100*100), nrow = 100, ncol = 100, byrow = TRUE)

for (i in c(1:100)) {
    A[i, i] <- i

    if (i + 1 < 101)  A[i, i + 1] <- 1/i
    if (i - 1 > 0)    A[i, i - 1] <- 1
}

# Define D.
d <- c(1:100)
d <- sin(d*pi/200)

# Solve Ax = d.
x <- solve(A, d)
x[1:5]
```

```
## [1] 0.005473329 0.010233988 0.010938907 0.012167224 0.012730871
```

## Part 2: For loops.

1. Write a function that uses a `for` loop to calculate the following with a sequence of $m$, and generate a plot for $m$ verses $E_m$. Avoid using a `for` loop, can you complete the same task?

$$E_m = 1 + \frac{1}{2} + \cdots + \frac{1}{2^m} - \log(2^m)$$

**Solution**:

```
# Using for loop.
E_m <- function(m) {
  res <- 1
  for (i in c(1:m)) {
    res <- res + 1/(2^i)
  }
  res <- res - log(2^m)

  return(res)
}

# Avoid using for loop.
E_m2 <- function(m) {
  res <- 1

  index <- c(1:m)
  denom <- 2^index
  res <- res + sum(1/denom)

  res <- res - log(2^m)

  return(res)
}

E_m(100)
```

```
## [1] -67.31472
```

```
E_m2(100)
```

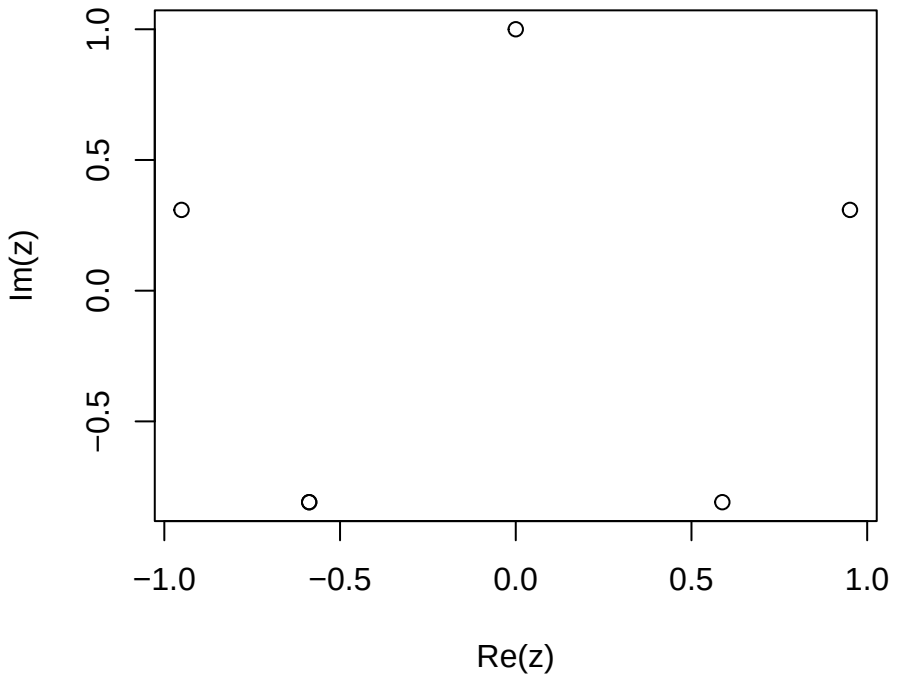
```
## [1] -67.31472
```

2. Let's draw a regular polygon of $n$ sides, with a horizontal bottom side, and the corners of the polygon staying on the unit circle. For given $n = 5$ and $r = 0.6$, start the first point $p_1 = (x_1, y_1)$ as a random number in $(-0.5, 0.5) \times (-0.5, 0.5)$, generate $10^4$ points interactively. In the $j$th iteration, we choose one corner $z_*$ of the polygon randomly and let $p_{j+1} = (x_{j+1}, y_{j+1})$ be the point on the line segment between $p_j$ and $z_*$, with the distance from $p_{j+1}$ to $p_j$ being $r$ times the distance from $z_j$ to $p_j$ and then draw all these points as dots in the $xy$ panel.

Hint: Complex numbers can be used to represent points in the $xy$ plane. The following script works for even or odd $n$, and the polygon always has a flat bottom.

```
n <- 5
t <- c(0:n) - 0.5
z <- exp(2i * pi * (t/n - 0.25))


plot(Re(z), Im(z))
```




**Solution**:

```
n_points <- 10000

# generate 10000 points
p <- rep(NA, n_points)

# starting from the 1st point from [-0.5, 0.5]x[-0.5, 0.5]
rand <- runif(1, min = 0, max = 1)
p[1] <- rand - 0.5 + 1i * (rand - 0.5)

# r x distance of d(z_j, p_j)
r <- 0.6

# set up the n sides polygon and n corners
```

```
n <- 5
t <- c(0:n) - 0.5
z <- exp(2i * pi * (t/n - 0.25))

# in the jth iteration, randomly choose one corner and update p(j+1)
for (j in c(2:n_points)) {
  # sample 1 corner
  zs <- z[sample(n, 1)]
  p[j] <- p[j-1] + r*(zs - p[j-1])
}

plot(Re(p), Im(p))
```