Segundo Reto: Análisis Numérico.

Álvarez Sánchez, Emanuel
alvarez_emanuel@javeriana.edu.co

Afanador Ochoa, Juan José
j.afanador@javeriana.edu.co

Burgos Melo, Diego Andrés
burgosmd@javeriana.edu.co

Barrera Martinez, Nicolai
barreram.n@javeriana.edu.co

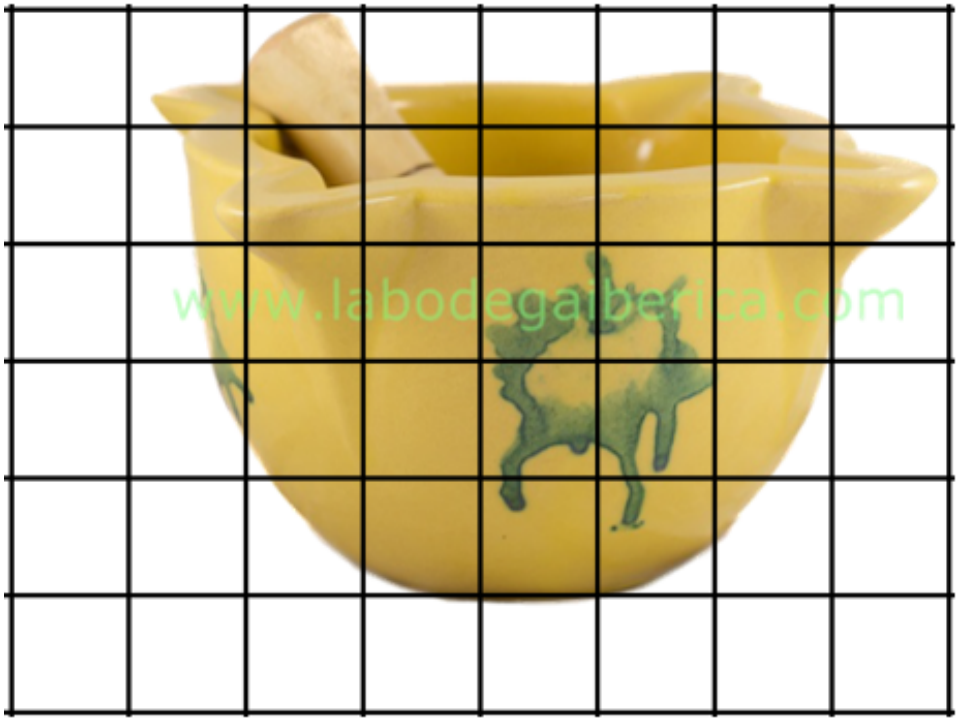
20 de abril de 2020

1. Introduction

A continuación veremos el respectivo documento del segundo reto de análisis numérico, donde podremos encontrar la descripción del problema escogido. El cual fue representar el mortero valenciano mediante la técnica de interpolación, usando puntos de regencia. también veremos el desarrollo de este mismo de forma analítica en GeoGebra y su desarrollo codificado en Rstudio. Para finalizar las conclusiones que nos deja los datos obtenidos y el desarrollo del problema.

2. Problema

El problema que fue escogido por el grupo, fue la representación de un mortero valenciano. El cual es un recipiente cóncavo, usado para realizar mezclas. Este mortero en específico posee cuatro salientes en sus bordes superiores, formando desde arriba una especie de estrella.



Mortero valenciano, dividido en cuadricula.

Teniendo en cuenta lo anterior se debía representar este mortero a partir de la unión de diversas superficies, para conseguir la representación en tres dimensiones del objeto. Usando intérpretes como RStudio o Phyton. Eligiendo de esta manera RStudio por sus técnicas interpolación como lo son "Bézier" y "BSplines", los cuales nos resultaban familiares.

3. Desarrollo

En cuanto al desarrollo, decidimos centrarnos por entender de manera optima en que consistía el método de Bézier, el cual genera curvas a partir de puntos o coordenadas (en el plano cartesiano X, Y) proporcionadas, generando figuras o superficies, que en su conjunto puede representar de forma matemática objetos en 2 dimensiones y 3 dimensiones.

3.1. Conocer las dimensiones del mortero valenciano.

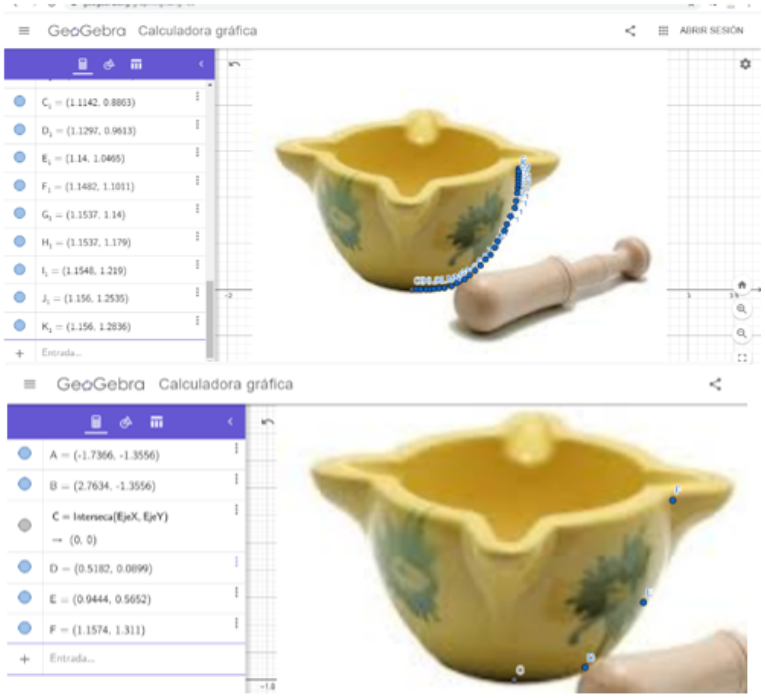
Imágenes de referencia del mortero valenciano. Nota: estas imágenes de referencias son necesarias para tener en cuenta el ancho, alto, diámetro, profundidad y perímetro del mortero.

Mortero valenciano, vista frente.

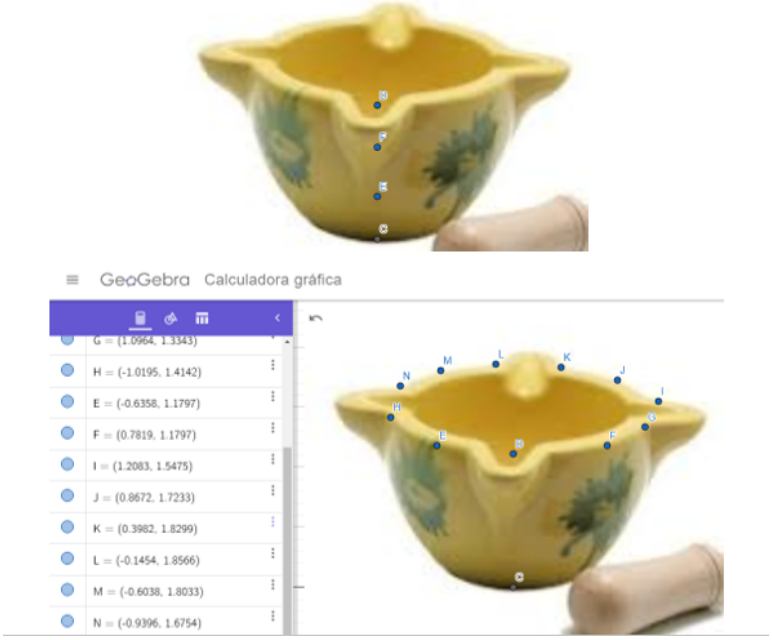
Mortero valenciano, vista superior.

3.2. Toma de puntos de forma experimental.

En la toma de puntos, fueron utilizados dos programas que nos ayudaron a comprender y obtener datos de la figura sobre un plano cartesiano. Los programas usados fueron: *Adobe Ilustrador: el cual nos ayudo a vectorizar la imagen y arrojar los la vista desde arriba, que posteriormente iba ser tratada. *GeoGebra: la cual nos ayudó a ceñir puntos en la imagen frontal sacando contorno y cerca de 100 coordenadas en X y Y, que conformaban el mortero valenciano.



Obteniendo puntos en Geogebra: contorno.

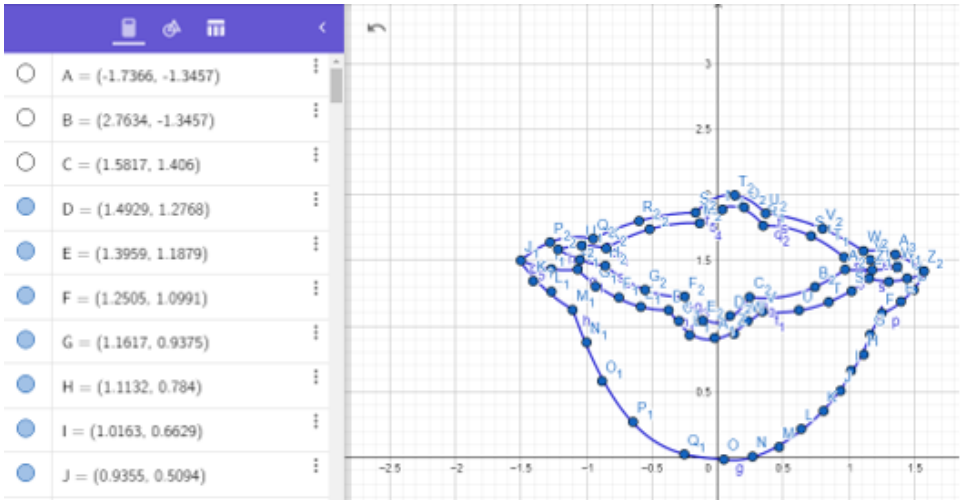


Obteniendo puntos en Geogebra: frente y superior.

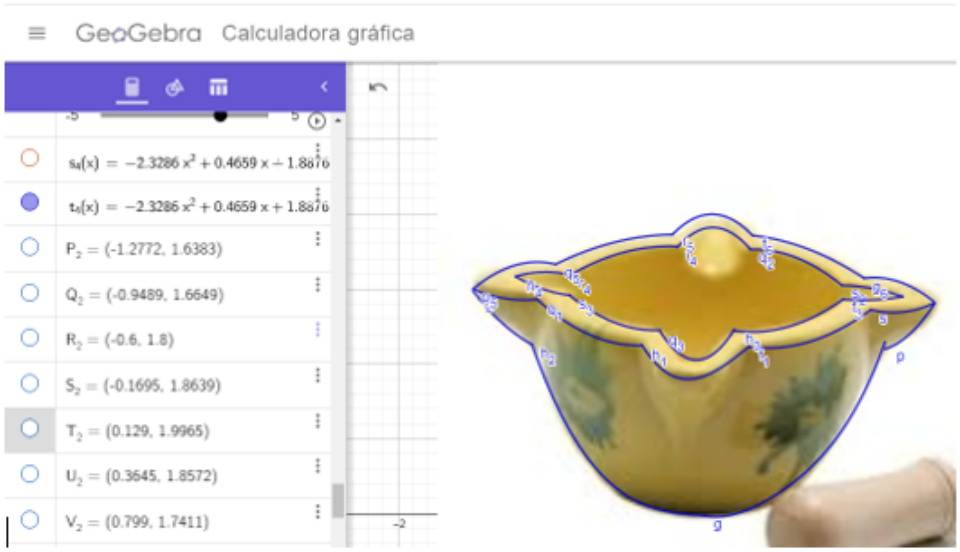
Estos puntos que fueron tomados son guardados y postriormente usados en la composición de curvas en RStudio.

3.3. Ajuste de puntos y creación de funciones.

En este paso los puntos tomados anteriormente se unen creando un ajuste polinómico, o una regresión de grado n (la que mas se ajuste), este proceso es facilitado por GeoGebra, usando la función .ªjustePolinómico(¡Arreglo de puntos¿,¡grado¿)ƨ un deslizador para acotar el mejor grado de la función a crear. Arrojando así, una función que se acote al diseño del mortero valenciano.



Mortero valenciano representado por curvas y puntos.



Mortero valenciano representado por curvas.

Finalmente se pudo comprender lo que se debe repricar en RStudio, a partir de los datos obtenidos en Geogebra y Adobe Ilustrator.

4. Implementacion de codigo

4.1. Descripcion de librerias requeridas para el uso del programa.

```
require(bezier)
require(gridBezier)
require(PolynomF)
require(rgl)
require(plot3D)
```

Las cuales representan el uso de interpolación como la función gridBezier o Bezier las cuales se utilizan para realizar interpolación entre varios puntos ya sean para curvas planas como para círculos, las cuales fueron implementadas con el fin de desarrollar el mortero, además de eso se quiere de funciones como rgl,o PolynomF para realizar las operaciones aritméticas o geométricas con respecto a los puntos, finalmente se utilizara la librería plot3D debido a que el desarrollo del mortero será efectuado en tres dimensiones.

4.2. Inicialización de datos y puntos.

Primeramente se inicializan los valores de x y y sobre los cuales vamos a tener un inicio de partida para realizar la interpolación, ya que conocemos que para realizar el método de Bezier es necesario realizar una interpolación entre 4 puntos los cuales se irán modificando a lo largo del programa.

```
p <- matrix(c(0,8,1,
              0.76,8,1,
              1.46,5.78,1,
              4.69,5.18,1,
              5.77,1.46,1,
              8,0.76,1,
              8,0,1), nrow=7, ncol=3, byrow=TRUE)

grid.newpage()
xsg <- BezierGrob(c(.0, .0, .076, 0.146), c(.8, .8, .8, .578),
gp=gpar(lwd=3), stepFn = nSteps(20))
xsg2 <- BezierGrob(c(.146, .146, .469, .577), c(.578, .578,
.518, .146),gp=gpar(lwd=3), stepFn = nSteps(60))
xsg3 <- BezierGrob(c(.577, .577, .8, .8), c(.146, .146, .076, .0),
gp=gpar(lwd=3), stepFn = nSteps(20))
trace <- BezierPoints(xsg)
trace2 <- BezierPoints(xsg2)
trace3 <- BezierPoints(xsg3)

valoresx=as.vector(trace$x)
valoresx2=as.vector(trace2$x)
valoresx3=as.vector(trace3$x)

valoresy=as.vector(trace$y)
valoresy2=as.vector(trace2$y)
valoresy3=as.vector(trace3$y)

datosx = c(valoresx,valoresx2,valoresx3)
datosy = c(valoresy,valoresy2,valoresy3)

grid.circle(trace$x, trace$y, default.units="inches",
r=unit(.5, "mm"))
```

```
grid.circle(trace2$x, trace2$y, default.units="inches",
r=unit(.5, "mm"))

grid.circle(trace3$x, trace3$y, default.units="inches",
r=unit(.5, "mm"))

X = datosx
Y = datosy
Z = c(rep(1,100))

for (i in 1:100) {
  X[i] = X[i]*2
  Y[i] = Y[i]*2
}
plot(X,Y, xlim=c(0,8), ylim=c(0,8) )
```

4.3. Creacion de la grafica externa.

Se crean los puntos finales y se crea un fondo externo con cuatro cuadrantes con el fin de implementar esos puntos para desarrollar el mortero valenciano.

```
xfinales1 = X[(contador-101):(contador-1)]
yfinales1 = Y[(contador-101):(contador-1)]
suelo = Z[contador-1]
X[contador-1]
posSuelo = contador-1

value = 0.02
aumento = seq(0,4000,100)
print(aumento)

contador2 = 1
X5 = datosx
Y5 = datosy
Z5 = c(rep(1,400))

for (j in aumento) {
  for (i in 1:100) {
```

```
    X5[i+j] = xfinales1[contador2]-value
    Y5[i+j] = yfinales1[contador2]-value
    Z5[i+j] = suelo
    contador2 = contador2 + 1
  }
  value = value + 0.02
  contador2 = 1
}
Xespejos1=-1*X5
Yespejos1=-1*Y5
```

Tambien se crea este fondo para tener una mejor visualización de los resultados del mortero con cuatro cuadrantes.Primeramente, para realizar el mortero se crea una gráfica 3D de cuatro cuadrantes con los valores externos, luego con sus puntos del suelo, mostrados a continuación:

```
#-Crear Grafica(externa) 3d con cuatro cuadrantes-#


plot3d(X, Y, Z, type = "p", col = "blue",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(Xespejo, Y, Z, type = "p", col = "blue",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) ,zlim = c(-3,3))

plot3d(Xespejo, Yespejo, Z, type = "p", col = "blue",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(X, Yespejo, Z, type = "p", col = "blue",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))


#--Graficar suelo(externo) 3d con cuatro cuadrantes---#

plot3d(X5, Y5, Z5, type = "p", col = "red",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))
```

```
plot3d(Xespejos1, Y5, Z5, type = "p", col = "red",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(Xespejos1, Yespejos1, Z5, type = "p", col = "red",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(X5, Yespejos1, Z5, type = "p", col = "red",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))
```

Luego de haber realizado las creaciones externas del mortero pasamos a implementar la interpolación con el fin de graficar los puntos internos completos con el fin de dar un mortero más completo y solidificado el cual no tenga ningún tipo de abertura u orificio además de eso se calcularán sus respectivos reflejos para generar una imagen 3D, se calcula la gráfica interna en la que podemos observar que se utiliza la interpolación con el método de Bezier para poder realizar los puntos internos, además de eso para tener una mejor realización utilizamos funciones como grid.circle() la cual es una función que crea un círculo gro() o un objeto que describe como un círculo el cual se utilizara como base para implementarse en el mortero

```
#----------GRFICA INTERNA----------#
xsg4 <- BezierGrob(c(.0, .0, .076, 0.146), c(.8, .8, .8,
.578),
gp=gpar(lwd=3), stepFn = nSteps(20))

xsg5 <- BezierGrob(c(.146, .146, .469, .577), c(.578, .578,
.518, .146),
gp=gpar(lwd=3), stepFn = nSteps(60))

xsg6 <- BezierGrob(c(.577, .577, .8, .8), c(.146, .146, .076,
.0),
gp=gpar(lwd=3), stepFn = nSteps(20))

trace4 <- BezierPoints(xsg4)
trace5 <- BezierPoints(xsg5)
trace6 <- BezierPoints(xsg6)
```

```
valoresx4=as.vector(trace4$x)
valoresx5=as.vector(trace5$x)
valoresx6=as.vector(trace6$x)

valoresy4=as.vector(trace4$y)
valoresy5=as.vector(trace5$y)
valoresy6=as.vector(trace6$y)

datosx2 = c(valoresx4,valoresx5,valoresx6)
datosy2 = c(valoresy4,valoresy5,valoresy6)

grid.circle(trace$x, trace4$y, default.units="inches",
r=unit(.5, "mm"))

grid.circle(trace2$x, trace5$y, default.units="inches",
r=unit(.5, "mm"))

grid.circle(trace3$x, trace6$y, default.units="inches",
r=unit(.5, "mm"))

X2 = datosx2
Y2 = datosy2
Z2 = c(rep(1,100))

for (i in 1:100) {
  X2[i] = X2[i]*1.6
  Y2[i] = Y2[i]*1.6
}
plot(X2,Y2, xlim=c(0,8), ylim=c(0,8) )
```

Utilizaremos estas variables para ir disminuyendo los valores de X Y y Z para generar un tipo de contenedor para posteriormente generar un mortero valenciano.

```
contador = length(X2) + 1
contador2 = 1
disminucion = 1-0.01
disminucion2 = 0.02
disminucion3 = 0.02
```

```
while(TRUE)
{
  if(X2[contador2]-disminucion3 >= 0)
  {
    X2[contador] = X2[contador2] - disminucion3
  }
  if(Y2[contador2] -disminucion2 >= 0)
  {
    Y2[contador] = Y2[contador2] -disminucion2
  }
  else
  {
    Y2[contador] = 0
    X2[contador]=X2[contador-1]
  }

  Z2[contador] = disminucion
  contador = contador + 1
  contador2 = contador2 + 1

  if(contador2 == 101)
  {
    contador2 = 1
    disminucion = disminucion - 0.01
    disminucion2 = disminucion2 + 0.02
    disminucion3 = disminucion3 + 0.02
  }
  if(contador == 8000)
  {
    cat("profundidad", Z2[contador-1], "\n")

    break;
  }
}

Xespejo2=-1*X2
Yespejo2=-1*Y2
```

De igual modo que para el proceso externo en el proceso interno realizaremos la creación del fondo, del suelo y la tapa de manera controlada y con

los cuatro cuadrantes respectivamente

```
#--Crear fondo(interno) con cuatro cuadrantes--#

xfinales = X2[(contador-101):(contador-1)]
yfinales = Y2[(contador-101):(contador-1)]

suelo = Z2[contador-1]
X2[contador-1]
posSuelo = contador-1


value = 0.02
aumento = seq(0,4000,100)
print(aumento)

contador2 = 1

X4 = datosx2
Y4 = datosy2
Z4 = c(rep(1,400))


for (j in aumento) {
  for (i in 1:100) {
    X4[i+j] = xfinales[contador2]-value
    Y4[i+j] = yfinales[contador2]-value
    Z4[i+j] = suelo
    contador2 = contador2 + 1
  }
  value = value + 0.02
  contador2 = 1
}

Xespejos=-1*X4
Yespejos=-1*Y4
```

Se realiza el grafico interno del mortero con el cual se va a conectar con las partes externas del mortero.

```
#--Crear grafica(interna) 3d con cuatro cuadrantes-#
```

```
plot3d(X2, Y2, Z2, type = "p", col = "blue",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) ,zlim = c(-3,3))

plot3d(Xespejo2, Y2, Z2, type = "p", col = "blue",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) ,zlim = c(-3,3))

plot3d(Xespejo2, Yespejo2, Z2, type = "p", col = "blue",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(X2, Yespejo2, Z2, type = "p", col = "blue",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))
```

Se grafica la parte interna del suelo con los mismos cuatro cuadrantes

```
#-Graficar suelo(interna) 3d con cuatro cuadrantes-#

plot3d(X4, Y4, Z4, type = "p", col = "green",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(Xespejos, Y4, 4, type = "p", col = "green",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) ,zlim = c(-3,3))

plot3d(Xespejos, Yespejos, Z4, type = "p", col = "green",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(X4, Yespejos, Z4, type = "p", col = "green",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))
```

Se realiza la TAPA del mortero la cual podemos ver como una boquilla que va a tener este mismo

```
#----------TAPA----------#
xsg4 <- BezierGrob(c(.0, .0, .076, 0.146), c(.8, .8, .8,
```

```
.578),gp=gpar(lwd=3), stepFn = nSteps(20))

xsg5 <- BezierGrob(c(.146, .146, .469, .577), c(.578, .578, .518,
.146),gp=gpar(lwd=3), stepFn = nSteps(60))

xsg6 <- BezierGrob(c(.577, .577, .8, .8), c(.146, .146, .076,
.0),gp=gpar(lwd=3), stepFn = nSteps(20))

trace4 <- BezierPoints(xsg4)
trace5 <- BezierPoints(xsg5)
trace6 <- BezierPoints(xsg6)

valoresx4=as.vector(trace4$x)
valoresx5=as.vector(trace5$x)
valoresx6=as.vector(trace6$x)

valoresy4=as.vector(trace4$y)
valoresy5=as.vector(trace5$y)
valoresy6=as.vector(trace6$y)

datosx2 = c(valoresx4,valoresx5,valoresx6)
datosy2 = c(valoresy4,valoresy5,valoresy6)

grid.circle(trace$x, trace$y, default.units="inches",
r=unit(.5, "mm"))

grid.circle(trace2$x, trace2$y, default.units="inches",
r=unit(.5, "mm"))

grid.circle(trace3$x, trace3$y, default.units="inches",
r=unit(.5, "mm"))


X3 = datosx2
Y3 = datosy2
Z3 = c(rep(1,400))

value = 1.7
aumento = seq(0,700,100)
print(aumento)
```

```
for (j in aumento) {
  for (i in 1:100) {
    X3[i+j] = datosx2[i]*value
    Y3[i+j] = datosy2[i]*value
  }
  value = value + 0.05
}

Xespejot=-1*X3
Yespejot=-1*Y3


plot(X3,Y3, xlim=c(0,8), ylim=c(0,8) )

plot3d(X3, Y3, Z3, type = "p", col = "red",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(Xespejot, Y3, Z3, type = "p", col = "red",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(Xespejot, Yespejot, Z3, type = "p", col = "red",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))

plot3d(X3, Yespejot, Z3, type = "p", col = "red",
       xlab = "x", ylab="y", zlab="z", xlim = c(-6,6),
       ylim = c(-6,6) , zlim = c(-3,3))
```

Para los efectos de precisicion es necesario tener en cuenta los calculos de posibles errores que se deben de tener calculados a continuacion, primero calcularemos el error que puede presentar cada punto por la tapa del mortero

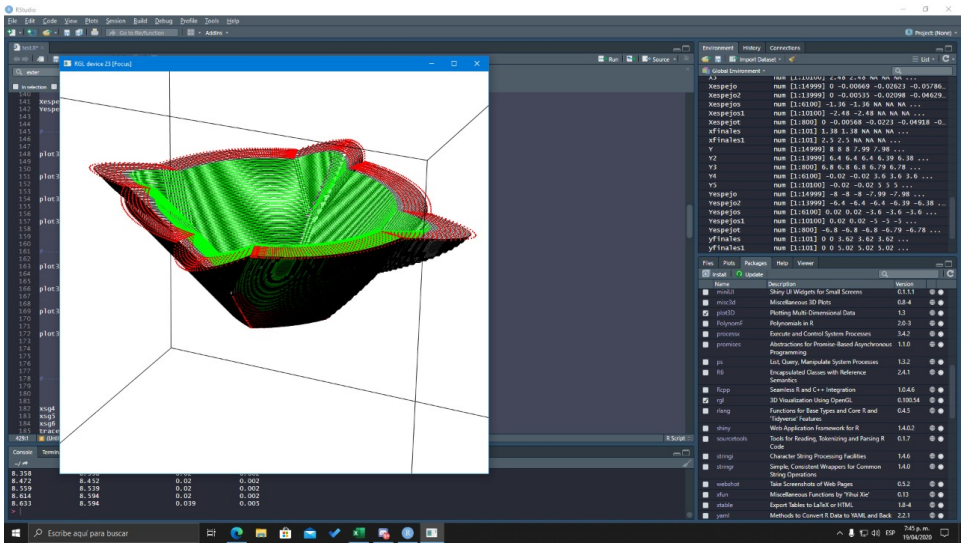
```
#--ERRORES-TAPA----#

auxX = X[1:100]

cat("X original","\t", "X tapa", "\t",
"\t", "error absoluto", "\t", "error relativo", "\n")
```

```
cont = 700
for (i in 1:100) {
  errorabs = round(abs(auxX[i]-X3[cont]),3)
  errorrel = round(errorabs/auxX[i],3)
  cat(round(auxX[i],3),"\t\t", round(X3[cont],3),
  "\t",  "\t\t", errorabs, "\t\t", errorrel, "\n")
  cont = cont+1
}
```

Para finalizar se calcula el error de la pared externa en cada uno de los puntos

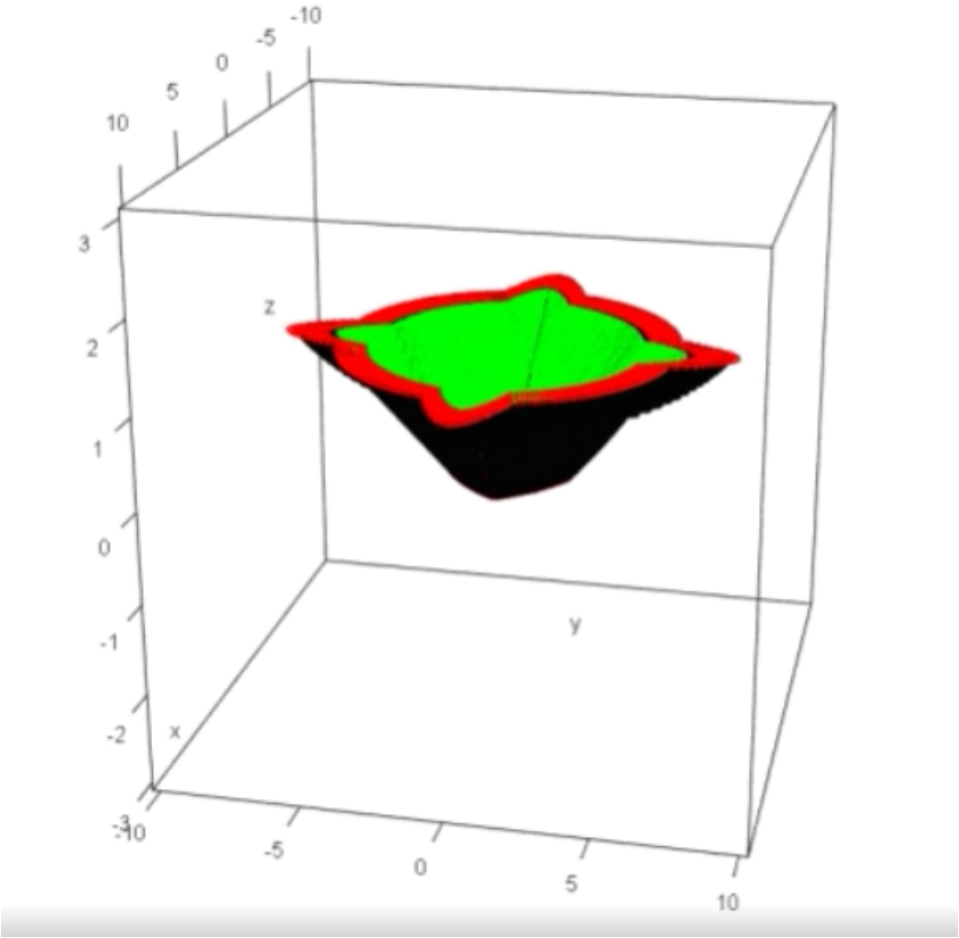
```
#---ERRORES PARED EXTERNA---#

auxX2 = X[1:100]


cat("X original","\t", "X pared", "\t",  "\t",
"error absoluto", "\t", "error relativo", "\n")

cont = 101
for (i in 1:100) {
  errorabs = round(abs(auxX2[i]-X[cont]),3)
  errorrel = round(errorabs/auxX2[i],3)
  cat(round(auxX2[i],3),"\t\t", round(X[cont],3),
  "\t",  "\t\t", errorabs, "\t\t", errorrel, "\n")
  cont = cont+1
}
```

Resultado despues de la compilacion y ejecucion del programa:



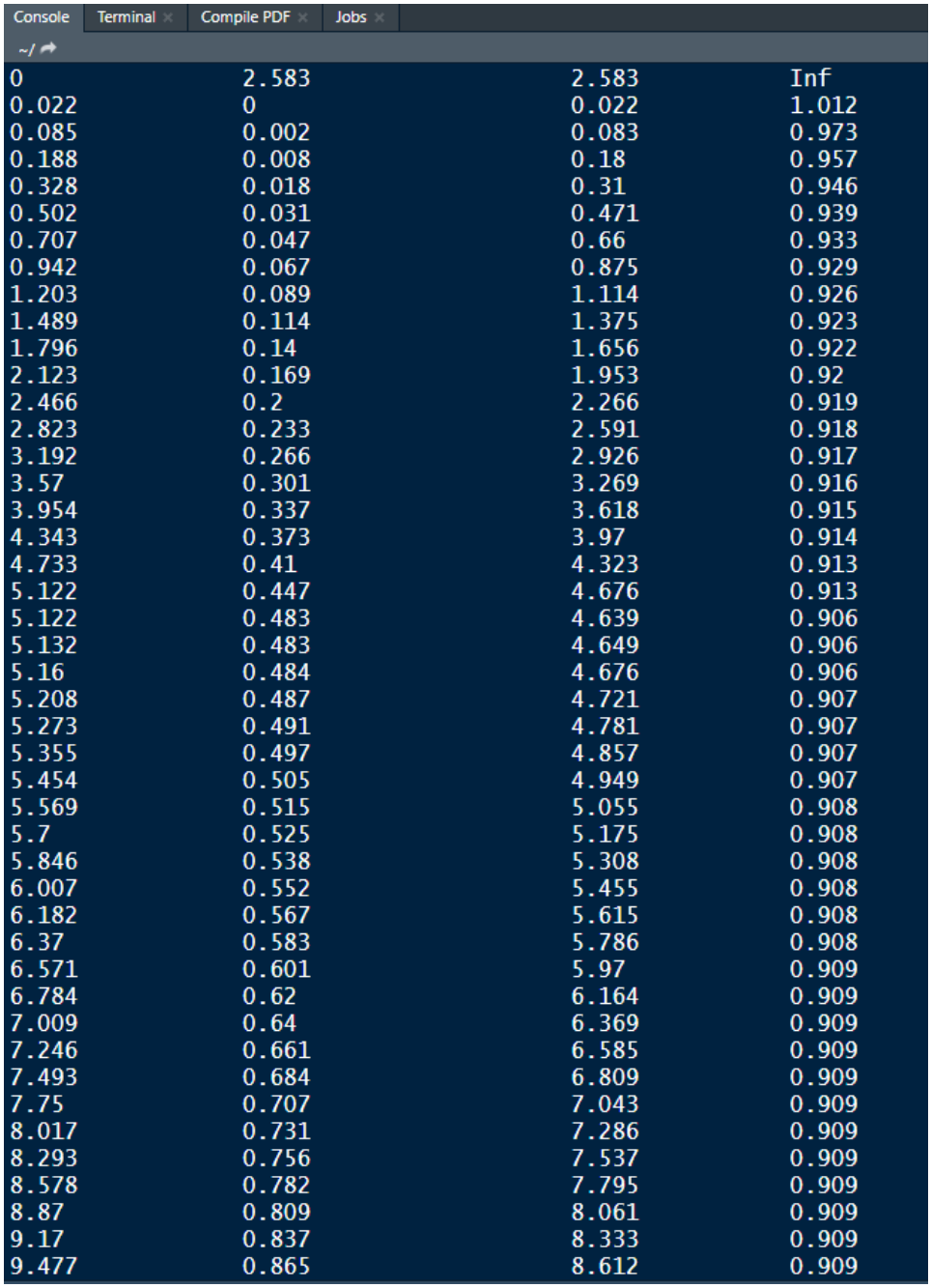
Mortero valenciano representado por curvas y puntos.

Mortero representado de manera 3D



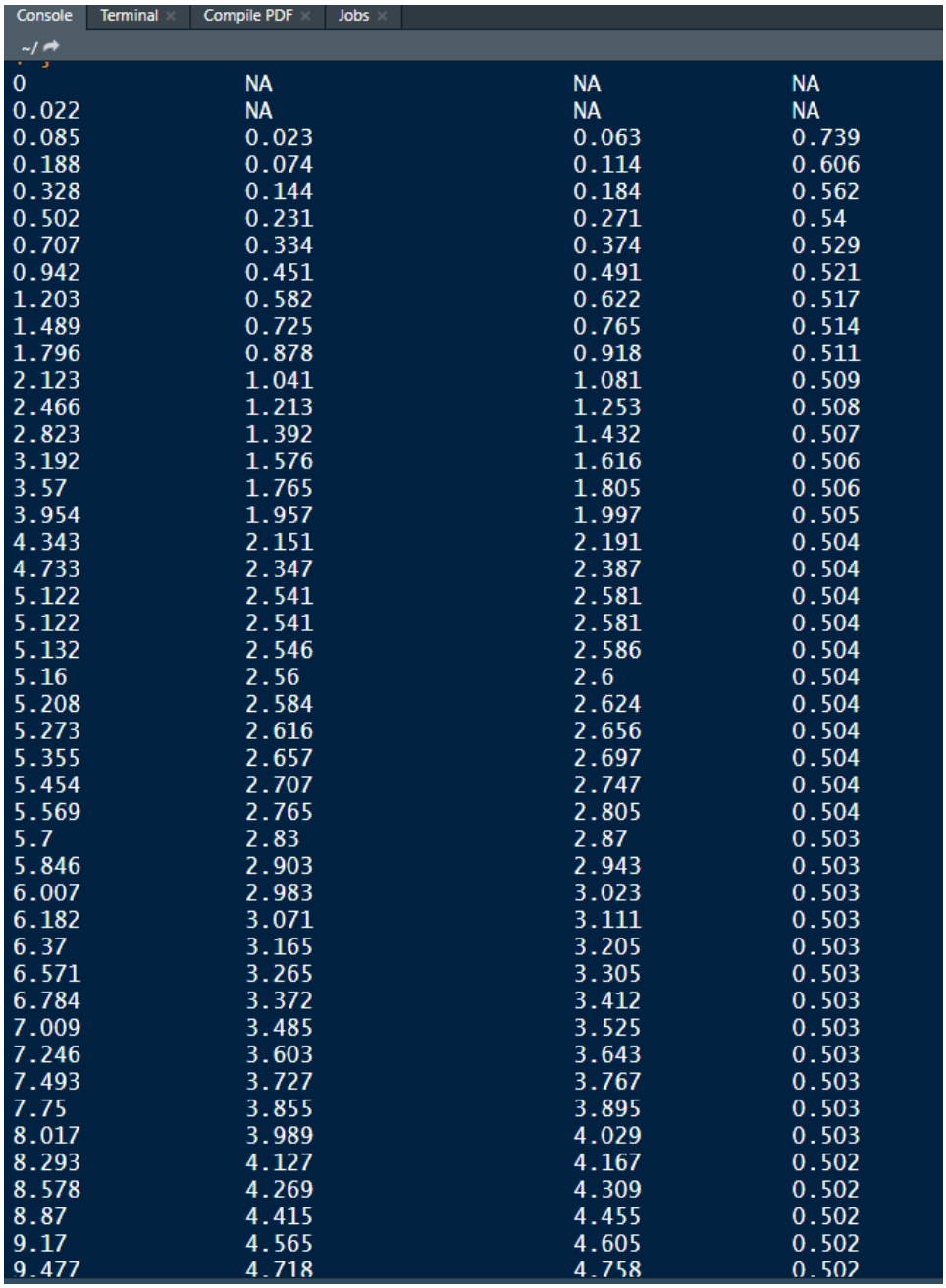
Mortero valenciano representado por curvas y puntos.

Finalmente realizamos las impresiones de las tablas de errores con los siguientes resultados



0	2.583	2.583	Inf
0.022	0	0.022	1.012
0.085	0.002	0.083	0.973
0.188	0.008	0.18	0.957
0.328	0.018	0.31	0.946
0.502	0.031	0.471	0.939
0.707	0.047	0.66	0.933
0.942	0.067	0.875	0.929
1.203	0.089	1.114	0.926
1.489	0.114	1.375	0.923
1.796	0.14	1.656	0.922
2.123	0.169	1.953	0.92
2.466	0.2	2.266	0.919
2.823	0.233	2.591	0.918
3.192	0.266	2.926	0.917
3.57	0.301	3.269	0.916
3.954	0.337	3.618	0.915
4.343	0.373	3.97	0.914
4.733	0.41	4.323	0.913
5.122	0.447	4.676	0.913
5.122	0.483	4.639	0.906
5.132	0.483	4.649	0.906
5.16	0.484	4.676	0.906
5.208	0.487	4.721	0.907
5.273	0.491	4.781	0.907
5.355	0.497	4.857	0.907
5.454	0.505	4.949	0.907
5.569	0.515	5.055	0.908
5.7	0.525	5.175	0.908
5.846	0.538	5.308	0.908
6.007	0.552	5.455	0.908
6.182	0.567	5.615	0.908
6.37	0.583	5.786	0.908
6.571	0.601	5.97	0.909
6.784	0.62	6.164	0.909
7.009	0.64	6.369	0.909
7.246	0.661	6.585	0.909
7.493	0.684	6.809	0.909
7.75	0.707	7.043	0.909
8.017	0.731	7.286	0.909
8.293	0.756	7.537	0.909
8.578	0.782	7.795	0.909
8.87	0.809	8.061	0.909
9.17	0.837	8.333	0.909
9.477	0.865	8.612	0.909

Tabla de error para la TAPA.



Console | Terminal | Compile PDF | Jobs

~/

0	NA	NA	NA
0.022	NA	NA	NA
0.085	0.023	0.063	0.739
0.188	0.074	0.114	0.606
0.328	0.144	0.184	0.562
0.502	0.231	0.271	0.54
0.707	0.334	0.374	0.529
0.942	0.451	0.491	0.521
1.203	0.582	0.622	0.517
1.489	0.725	0.765	0.514
1.796	0.878	0.918	0.511
2.123	1.041	1.081	0.509
2.466	1.213	1.253	0.508
2.823	1.392	1.432	0.507
3.192	1.576	1.616	0.506
3.57	1.765	1.805	0.506
3.954	1.957	1.997	0.505
4.343	2.151	2.191	0.504
4.733	2.347	2.387	0.504
5.122	2.541	2.581	0.504
5.122	2.541	2.581	0.504
5.132	2.546	2.586	0.504
5.16	2.56	2.6	0.504
5.208	2.584	2.624	0.504
5.273	2.616	2.656	0.504
5.355	2.657	2.697	0.504
5.454	2.707	2.747	0.504
5.569	2.765	2.805	0.504
5.7	2.83	2.87	0.503
5.846	2.903	2.943	0.503
6.007	2.983	3.023	0.503
6.182	3.071	3.111	0.503
6.37	3.165	3.205	0.503
6.571	3.265	3.305	0.503
6.784	3.372	3.412	0.503
7.009	3.485	3.525	0.503
7.246	3.603	3.643	0.503
7.493	3.727	3.767	0.503
7.75	3.855	3.895	0.503
8.017	3.989	4.029	0.503
8.293	4.127	4.167	0.502
8.578	4.269	4.309	0.502
8.87	4.415	4.455	0.502
9.17	4.565	4.605	0.502
9.477	4.718	4.758	0.502

Tabla de error para la pared externa.

5. Conclusiones

En conclusión, podemos observar que la interpolación es una técnica muy usada en la actualidad en diversas áreas de estudio, ya que nos brinda la posibilidad de modelar un objeto de la vida real, convirtiéndolo en un modelo matemático, hecho a partir de coordenadas cartesianas en los ejes X, Y, Z. Estos modelos traen consigo muchos beneficios, ya que, al podernos manipular de forma libre, teniendo en cuenta la primera aproximación una de las tantas aplicaciones puede ser el escalamiento del objeto y la simulación de este a dicha escala, cosa que sería muy costosa si se hace en la vida real.

Además de eso con respecto a los resultados pudimos obtener con respecto a la tabla de errores es la evidencia de una proporcionalidad directa ya que a medida que los puntos van aumentando su valor así mismo va aumentando la relación con el error, por lo que decimos que al momento de querer generar un mortero mas grande vamos a tener un problema de precisión debido a la incertidumbre que generan los errores, por ello concluimos el error es directamente proporcional con tanto la cantidad de puntos y con la distancia que hay entre un punto y otro.

Por otra parte, podemos concluir que, las curvas de Bézier nos ayudan a modelar de manera muy fácil e intuitiva el mortero valenciano en RStudio, a partir de 4 puntos base que nos generan la curva y que el conjunto de estas curvas nos genera a su vez superficies creando objetos en 3 dimensiones.