Weekly Assignment 3

RC

9/8/2021

```
import sympy as sy
import sympy.vector as sv

N = sv.CoordSys3D('N')
t = sy.Symbol('t', real = True)
```

Question 1

```
r = 2*sy.cos(t)*N.i+4/sy.sqrt(t)*N.j+5/(t-3)*N.k
print(f"r(t) = {r}")
```

```
## r(t) = (2*cos(t))*N.i + (4/sqrt(t))*N.j + (5/(t - 3))*N.k
```

$t > 0, t \neq 3$

$r(t)$ defined on $[-2, 2]\hat{i} + (0, oo)\hat{j} + (-oo, oo)\hat{k}$

Question 2

Part A

```
r = 2*sy.sin(t)*N.i+sy.exp(5*t)*N.j-4*sy.cos(3*t)*N.k
print(f"r(t) = {r}")
```

```
## r(t) = (2*sin(t))*N.i + (exp(5*t))*N.j + (-4*cos(3*t))*N.k
```

```
print(f"r(0) = {r.subs(t, 0).evalf()}")
```

```
## r(0) = 1.0*N.j + (-4.0)*N.k
```

Part B

```
print(f"r'(t) = {sy.diff(r, t)}")
```

```
## r'(t) = (2*cos(t))*N.i + (5*exp(5*t))*N.j + (12*sin(3*t))*N.k
```

Part C

```
print(f"r''(t) = {sy.diff(sy.diff(r, t), t)}")
```

```
## r''(t) = (-2*sin(t))*N.i + (25*exp(5*t))*N.j + (36*cos(3*t))*N.k
```

Part D

```
print(f"Integral of r dt = {sy.integrate(r, t).doit()}")
```

```
## Integral of r dt = (-2*cos(t))*N.i + (exp(5*t)/5)*N.j + (-4*sin(3*t)/3)*N.k
```

Don't forget the integration constants.

$$\int r(t)dt = (-2cos(t) + c_1)\hat{i} + (\frac{1}{5}e^{5t} + c_2)\hat{j} + (\frac{-4}{3}sin(3t) + c_3)\hat{k}$$

where $c_1, \ \ c_2, \ \ c_3$ are integration constants.

Question 3

```
r_prime = 3*t*N.i+(2-4*t)*N.j+6*t**2*N.k
print(f"r'(t) = {r_prime}")
```

```
## r'(t) = 3*t*N.i + (2 - 4*t)*N.j + 6*t**2*N.k
```

```
r_at_2 = 3*N.i + N.j -2*N.k
print(f"r(2) = {r_at_2}")
```

```
## r(2) = 3*N.i + N.j + (-2)*N.k
```

```
r = sy.integrate(r_prime, t).doit() # constants not included
print(f"r(t) without constants = {r}")
```

```
## r(t) without constants = 3*t**2/2*N.i + (-2*t**2 + 2*t)*N.j + 2*t**3*N.k
```

```
constants = r_at_2 - r.subs(t, 2)
print(f"r(t) = {constants + r}")
```

```
## r(t) = (3*t**2/2 - 3)*N.i + (-2*t**2 + 2*t + 5)*N.j + (2*t**3 - 18)*N.k
```

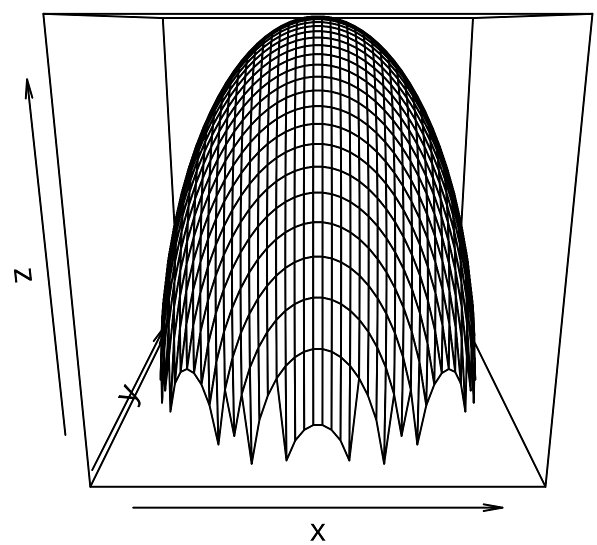
Question 4

Part A

A Sphere and a Paraboloid

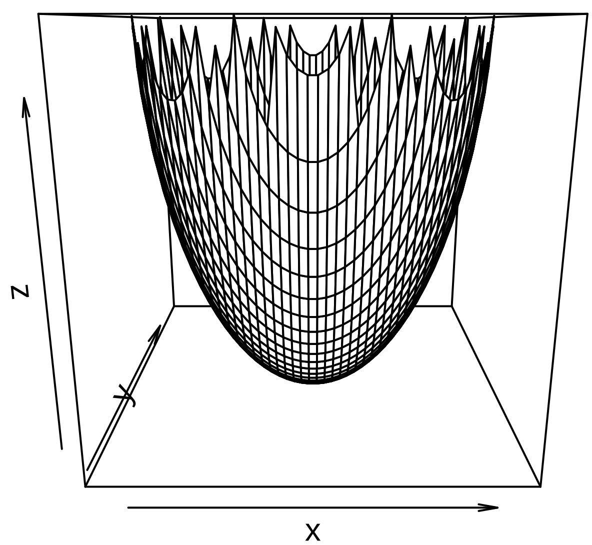
```
f <- function(x,y) ifelse((20 - x^2 - y^2) >= 0, sqrt(20 - x^2 - y^2), NA)
g <- function(x,y) ifelse((20 - x^2 - y^2) >= 0, -sqrt(20 - x^2 - y^2), NA)
h <- function(x,y) x^2 + y^2
xseq <- seq(-5, 5, length = 50)
yseq <- seq(-5, 5, length = 50)
persp(x = xseq, y = yseq,
      z = matrix(f(rep(xseq, each=50), rep(yseq, times=50)), nrow=50, ncol=50),
      xlab="x", ylab="y", zlab="z")
```

```
## Warning in sqrt(20 - x^2 - y^2): NaNs produced
```

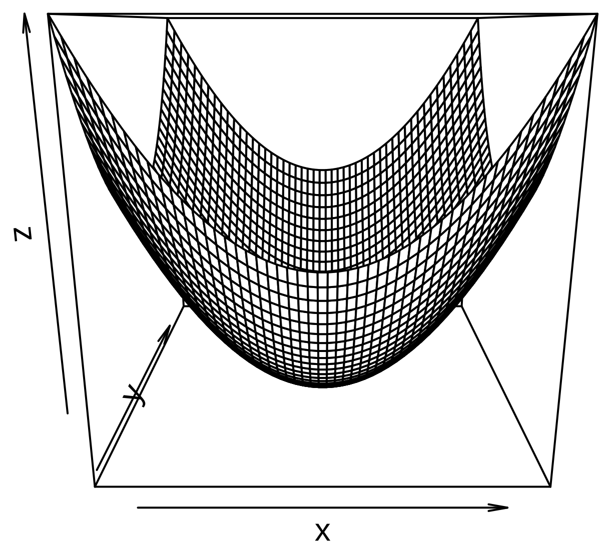



```
persp(x = xseq, y = yseq,
      z = matrix(g(rep(xseq, each=50), rep(yseq, times=50)), nrow=50, ncol=50),
      xlab="x", ylab="y", zlab="z")
```

```
## Warning in sqrt(20 - x^2 - y^2): NaNs produced
```



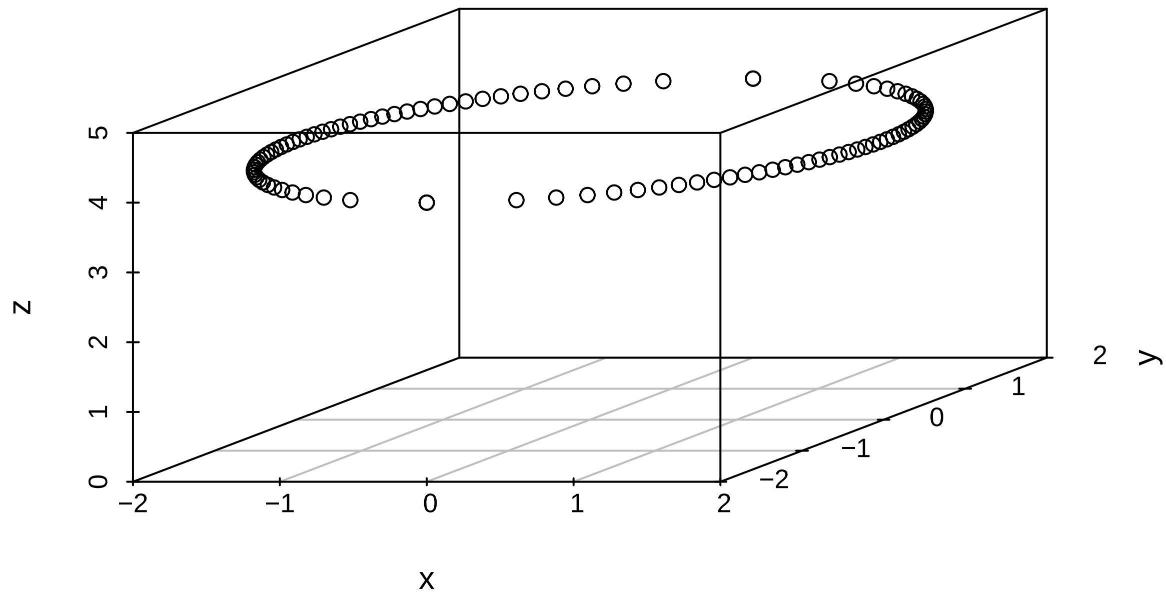

```
persp(x = xseq, y = yseq,
      z = matrix(h(rep(xseq, each=50), rep(yseq, times=50)), nrow=50, ncol=50),
      xlab="x", ylab="y", zlab="z")
```



Part B

A circle on the surface of the sphere in a plane parallel to the x-y plane

```
# substitute z into the other equation
w <- function(x, y) (x^2 + y^2 + (x^2 + y^2)^2 - 20)^2
# solve for positive x given y
xs <- sapply(seq(-2, 2, length=50), function(yy) optimize(w, c(0, 5), y=yy)$minimum)
# computer z given x and y pairs
zs <- h(xs, seq(-2, 2, length=50))
# plot with both positive and negative x's
scatterplot3d::scatterplot3d(c(xs, -xs),
                             c(seq(-2, 2, length=50),
                               seq(-2, 2, length=50)),
                             c(zs, zs), zlim = c(0, 5),
                             xlab = "x", ylab = "y", zlab = "z")
```



Part C

If $x = 2cos(t)$ and if the circle is in a plane of constant z on the sphere, then it is reasonable to assume a transform to cylindrical coordinates will work.

$$x = rcos(\theta) = 2cos(t)$$

$$r = 2, \quad \theta = t$$

$$y = rsin(\theta) = 2sin(t)$$

$$z_{rect} = z_{cyl} = x^2 + y^2 = 4cos^2(t) + 4sin^2(t) = 4$$

Check

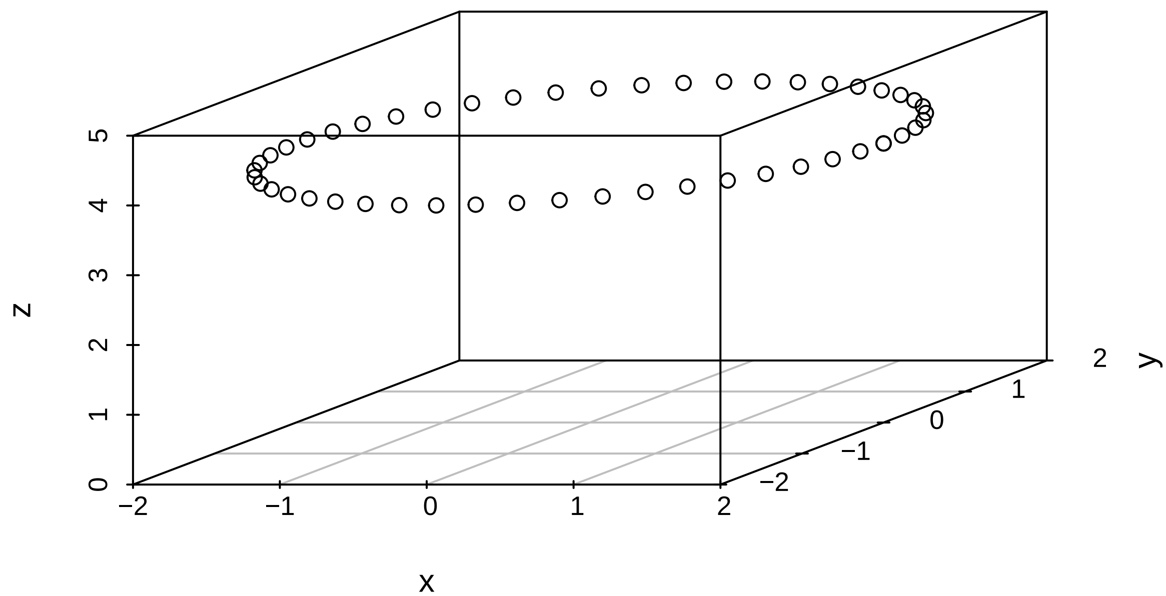
$$x^2 + y^2 + z^2 = 4cos^2(t) + 4sin^2(t) + 4^2 = 4 + 4^2 = 20$$

Therefore

$$l(t) = 2cos(t)\hat{i} + 2sin(t)\hat{j} + 4\hat{k}$$

Plot this line

```r
el <- function(t) c(2*cos(t), 2*sin(t), 4)
X <- t(sapply(seq(0, 2*pi, length = 50), el))
scatterplot3d::scatterplot3d(x = X[,1], y = X[,2], z = X[,3], zlim=c(0,5),
                             zlab = "z", xlab = "x", ylab = "y")
```



Question 5

```python
r = t**2*N.i + 9*t*N.j + t**3*N.k
print(f"r(t) = {r}")
```

```
## r(t) = t**2*N.i + 9*t*N.j + t**3*N.k
```

```python
r_prime = sy.diff(r, t)
print(f"r'(t) = {r_prime}")
```

```
## r'(t) = 2*t*N.i + 9*N.j + 3*t**2*N.k
```

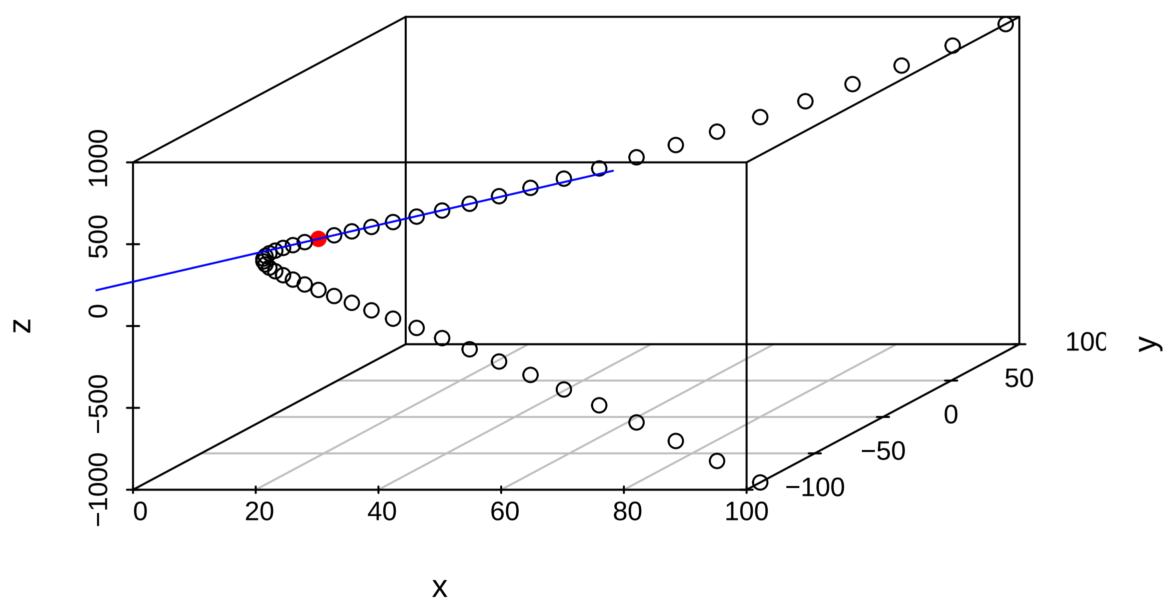
```python
l = r.subs(t, 2) + r_prime.subs(t, 2)*t
print(f"l(t) = r(2) + t*r'(2) = {l}")
```

```
## l(t) = r(2) + t*r'(2) = (4*t + 4)*N.i + (9*t + 18)*N.j + (12*t + 8)*N.k
```

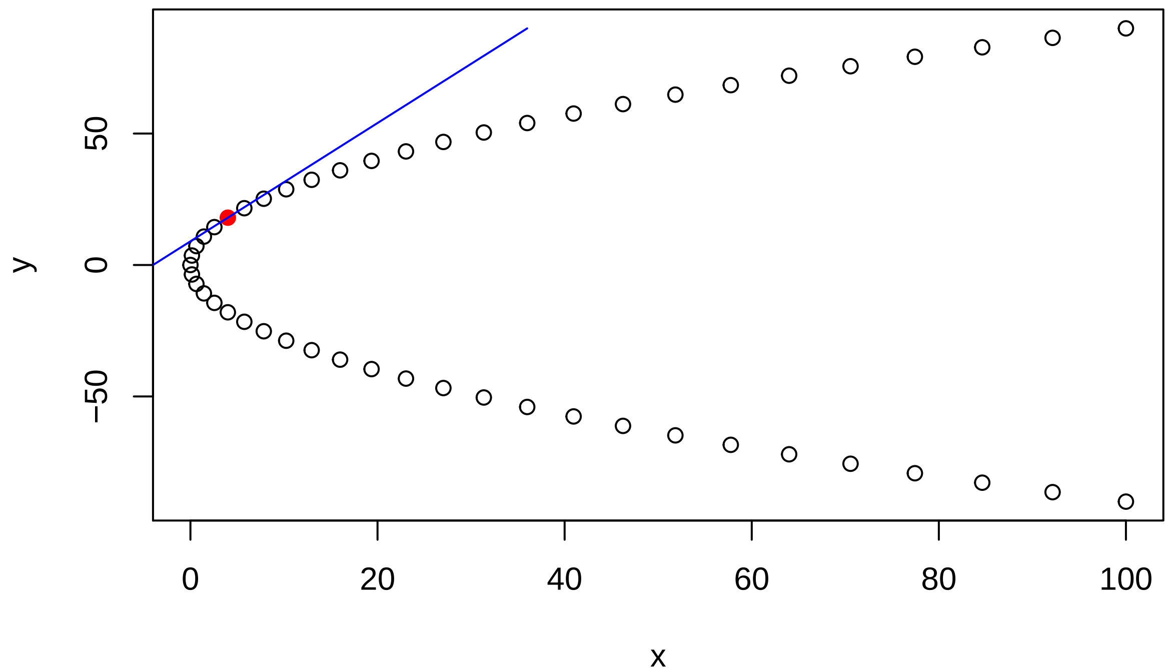
Check with plots

```r
X <- data.frame(tr = seq(-10, 10, length = 51))
X$col <- ifelse(X$tr == 2, "red", "black")
X$pch <- ifelse(X$tr == 2, 19, 1)
X$x <- X$tr^2
```

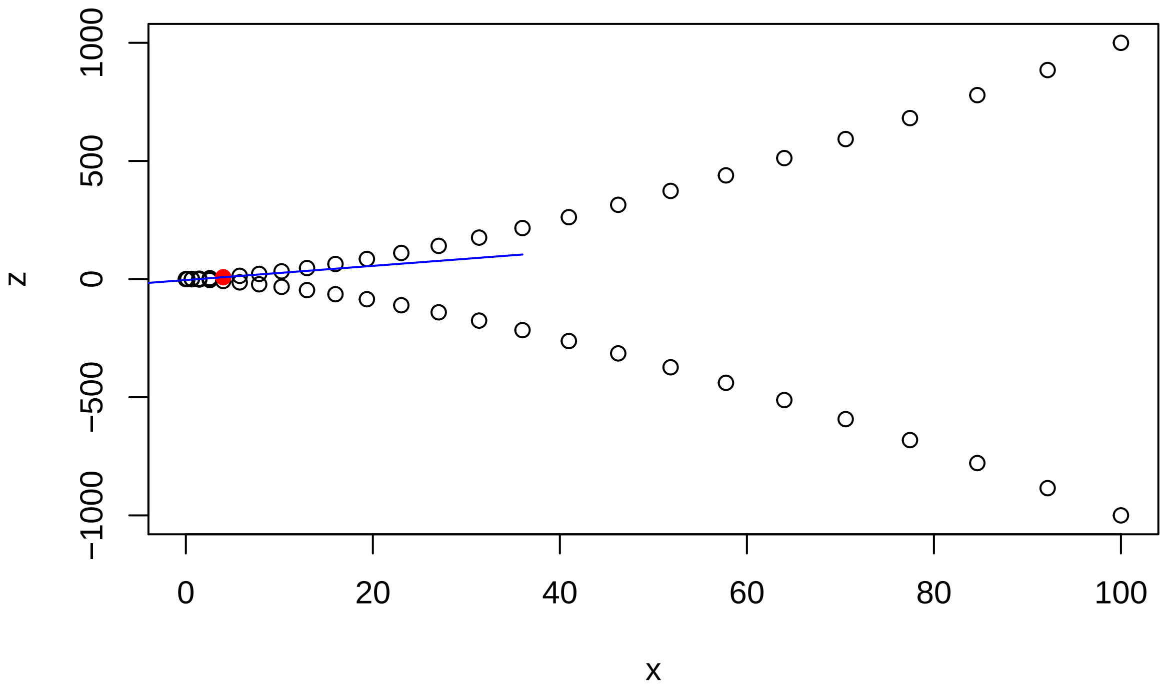
```
X$y <- 9*X$tr
X$z <- X$tr^3

s3d <- with(X, scatterplot3d::scatterplot3d(x=x, y=y, z=z,
                                           xlab="x", ylab="y", zlab="z",
                                           color = X$col, pch = X$pch))

X$tl <- seq(-12, 8, length = 51)
stopifnot(X$tl[X$tr == 2] == 0)

X$xl <- 4*X$tl + 4
X$yl <- 9*X$tl + 18
X$zl <- 12*X$tl + 8

s3d$points3d(X$xl, X$yl, X$zl, type = "l", col = "blue")
```




```
par(mar = c(5,4,4,2) + 0.1)
plot(X$x, X$y, col = X$col, pch = X$pch, xlab = "x", ylab = "y")
lines(X$xl, X$yl, col = "blue")
```




```r
plot(X$x, X$z, col = X$col, pch = X$pch, xlab = "x", ylab = "z")
lines(X$xl, X$zl, col = "blue")
```



Question 6

```
r = sy.cos(4*t)*N.i+sy.sin(4*t)*N.j+sy.exp(3*t)*N.k
print(f"r(t) = {r}")
```

```
## r(t) = (cos(4*t))*N.i + (sin(4*t))*N.j + (exp(3*t))*N.k
```

```
r_prime = sy.diff(r, t)
print(f"r'(t) = {r_prime}")
```

```
## r'(t) = (-4*sin(4*t))*N.i + (4*cos(4*t))*N.j + (3*exp(3*t))*N.k
```

```
s = sy.integrate(sy.sqrt(r_prime.dot(r_prime)), (t, 0, 3))
print(f"Numeric integration result, s = {s.evalf()}")
```

```
## Numeric integration result, s = 8102.88196759600
```

$$s = \int_0^3 \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2} dt$$

$$s = \int_0^3 \sqrt{16sin^2(4t) + 16cos^2(4t) + 9e^{6t}} dt$$

$$s = \int_0^3 \sqrt{16 + 9e^{6t}} dt$$

Question 7

Part A

```
v_0,h,a,g = sy.symbols('v_0 h a g')
z = h + v_0*sy.sin(a)*t + 0.5*g*t**2
print(f"vertical position, z(t) = {z}")
```

```
## vertical position, z(t) = 0.5*g*t**2 + h + t*v_0*sin(a)
```

```
x = v_0*sy.cos(a)*t
print(f"horizontal position, x(t) = {x}")
```

```
## horizontal position, x(t) = t*v_0*cos(a)
```

```
y = 0
print(f"Into-the-page position, y(t) = 0")
```

```
## Into-the-page position, y(t) = 0
```

Part B

```
print(f"solve z(t) for t, t = {sy.solve(z, t)}")
```

```
## solve z(t) for t, t = [(-v_0*sin(a) + 1.4142135623731*sqrt(-g*h + 0.5*v_0**2*sin(a)**2))/g, -(v_0*si
```

```
t_end1 = sy.solve(z, t)[0].subs(v_0,1800).subs(a,34/180*sy.pi).subs(g, -32.1741).subs(h, 500).evalf()
t_end2 = sy.solve(z, t)[1].subs(v_0,1800).subs(a,34/180*sy.pi).subs(g, -32.1741).subs(h, 500).evalf()
print(f"substitute in for the constants, t = {t_end1}, {t_end2}")
```

```
## substitute in for the constants, t = -0.492865294378723, 63.0616536208490
```

```
print("choose the positive time")
```

```
## choose the positive time
```

```
x_f = 0 + v_0*sy.cos(a)*t_end2
print(f"substitute in for x(t), x(landing time) = {x_f.subs(v_0, 1800).subs(a,34/180*sy.pi).evalf()}")
```

```
## substitute in for x(t), x(landing time) = 94104.8644304440
```

symbolically,

$$t = \frac{-v_0 sin(a) \pm \sqrt{v_0^2 sin^2(a) - 4(0.5g)(h)}}{2(0.5g)}$$

Question 8

Part A

```
r = (2*t**3+3)*N.i + (4*t**2-2)*N.j + (t**2+2*t)*N.k
print(f"r(t) = {r}")
```

```
## r(t) = (2*t**3 + 3)*N.i + (4*t**2 - 2)*N.j + (t**2 + 2*t)*N.k
```

```
print(f"r(0) = {r.subs(t, 0)}")
```

```
## r(0) = 3*N.i + (-2)*N.j
```

Part B

```
print(f"r(2) = {r.subs(t, 2)}")
```

```
## r(2) = 19*N.i + 14*N.j + 8*N.k
```

Part C

```
r_prime = sy.diff(r, t)
print(f"r'(t) = {r_prime}")
```

```
## r'(t) = 6*t**2*N.i + 8*t*N.j + (2*t + 2)*N.k
```

Part D

```
#s = sy.integrate(sy.sqrt(r_prime.dot(r_prime)).simplify(), t)
#s
s = sy.integrate(sy.sqrt(r_prime.dot(r_prime)), (t, 0 ,2))
print(f"s = {s.evalf()}")
```

```
## s = 24.7802488391474
```

Part E

```
print(f"distance from t=0 to t=2 = {(r.subs(t,2) - r.subs(t,0)).magnitude()}")
```

```
## distance from t=0 to t=2 = 24
```

Part F

Distance along curved path is longer than straight line path.