Term Project

Engineering Analysis and Computation

Dr. Heba Resk

ENGR 3202-01

*December 12, 2019*

**Authors:**

Karim Abdel Hamid *900 143 463*

Khaled Soliman *900 162 031*

# Question 1

## Project Description

The main problem has to do with taking all forces and integrating using various methods to find the velocity.

## Background on Numerical Methods

## Equations Programmed

Newton-Raphson:

## Output

|  |  |  |
| --- | --- | --- |
| Terminal Speed | Iterations Taken | Error |
| 3.5504 | 4 | 7.5798 x 10-09 |

Analytical:

# Question 5

## Project Description

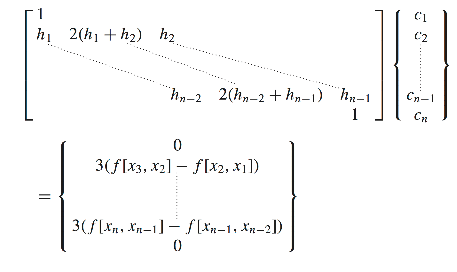
There is a correlation between ApCD and v, but not a direct formula. Using cubic interpolation, we can draw a curve through the points, and find another point along it (specifically v=16m/s).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 4 | 5.5 | 7 | 8 | 9.5 | 11 | 12.5 | 14 | 15.5 |
| f(x) | 0.435 | 0.3975 | 0.395 | 0.3825 | 0.38 | 0.37 | 0.3775 | 0.38 | 0.385 |
| x | 16.5 | 18 | 19.5 | 21 | 22.5 | 23.5 | 25 | 26.5 | 28 |
| f(x) | 0.3825 | 0.385 | 0.39 | 0.3975 | 0.395 | 0.41 | 0.4075 | 0.425 | 0.4275 |

## Background on Numerical Methods

## Equations Programmed

Cubic Spline Algorithm



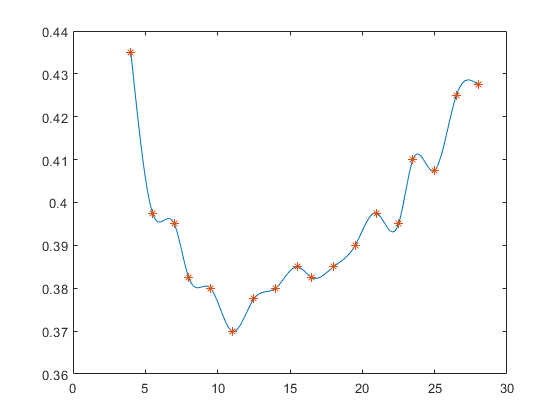
TDMA Algorithm

## Problems

As always, round-off floating point errors can be a concern. The main issue likely comes from the accuracy of the initial points: no values were given so I read them by eye.

## Output

Evaluating cubic at v=16m/s spline gives an ApCD of: **0.3839**



# Appendix

## TDMA.m

function [x] = TDMA(e,f,g,r)

n = size(f, 1);

x = zeros(n, 1);

for k=2:n

e(k) = e(k) / f(k-1);

f(k) = f(k) - e(k)\*g(k-1);

end

for k=2:n

r(k) = r(k) - e(k) \* r(k-1);

end

x(n) = r(n) / f(n);

for k=n-1:-1:1

x(k) = (r(k) - g(k) \* x(k+1)) / f(k);

end

end

## Cubic.m

function [a,b,c,d] = Cubic(X, F)

n = size(X, 2);

e = zeros(n, 1);

f = zeros(n, 1);

g = zeros(n-1, 1);

r = zeros(n, 1);

f(1) = 1;

f(n) = 1;

w = @(y, z) (F(y) - F(z)) / (X(y) - X(z));

h = zeros(n-1);

for i = 1:n-1

h(i) = X(i + 1) - X(i);

end

for i = 2:n-1

e(i) = h(i-1);

f(i) = 2 \* (h(i - 1) + h(i));

g(i) = h(i);

r(i) = 3 \* (w(i+1, i) - w(i, i-1));

end

c = TDMA(e, f, g, r);

a = F;

b = zeros(n-1, 1);

d = zeros(n-1, 1);

for i = 1:n-1

d(i) = (c(i+1)-c(i)) / (3\*h(i));

b(i) = ((F(i+1)-F(i))/h(i)) - (2 \* c(i)+c(i+1))\*h(i)/3;

end

end

## EvalCubic.m

function [f] = EvalCubic(x, X, a, b, c, d)

n = size(X, 2);

m = n-1;

for i=2:n

if (x < X(i))

m = i-1;

break;

end

end

l = x - X(m);

f = a(m) + b(m)\*l + c(m)\*(l.^2) + d(m)\*(l.^3);

end

## NewtonRaphson.m

*Note: Loosely based off Fixed-Point-Iteration code in Lecture14, Slide9*

function [out, iter, ea] = NewtonRaphson(f, fd, x0, es, imax)

xr = x0;

iter = 0;

ea = 100.0;

while (true)

xrold = xr;

xr = xrold - f(xrold) / fd(xrold);

iter = iter + 1;

if (xr ~= 0)

ea = 100.0 \* abs((xr - xrold) / xr);

end

if (ea < es || iter >= imax)

break;

end

end

out = xr;

end

## Q1.m

diary Q1.txt

imax = 100;

es = 0.001;

v0 = 0;

g=9.81;

Cr = 40;

grade = 0.0925;

theta = atan(grade);

ct = cos(theta);

st = sin(theta);

Pavg = 440;

m = 90;

row = 1.225; % kg/m3 at sea level and 15 degrees C

ApCd= 0.36;

k = 0.5 \* row \* ApCd;

f = @(v)(-k\*v.^3-v\*Cr\*ct- v\*m\*g\*st+Pavg);

fd = @(v) (-3\*v\*v\*k - Cr\*ct - m\*g\*st);

[out, iter, ea] = NewtonRaphson(f, fd, v0, es, imax)

diary off

## Q5.m

diary Q5.txt

x = 16.0;

X = [4, 5.5, 7, 8,9.5, 11, 12.5, 14, 15.5, 16.5, 18, 19.5, 21, 22.5, 23.5, 25, 26.5, 28];

F = [0.435, 0.3975, 0.395, 0.3825, 0.38, 0.37, 0.3775, 0.38, 0.385, 0.3825, 0.385, 0.39, 0.3975, 0.395, 0.41, 0.4075, 0.425, 0.4275];

[a,b,c,d] = Cubic(X, F);

%f = EvalCubic(x, X, a, b, c, d);

q = 4:0.05:28;

[Q] = arrayfun(@(p)EvalCubic(p, X, a, b, c, d), q);

plot(q, Q, X, F, '\*');

hold on

disp("Evaluating cubic spline gives: ");

disp(EvalCubic(x, X, a, b, c, d));

diary off