For this assignment I have all the output and code in the same console application. I forgot to check with the instructor on if this was ok, but the output and code for all programming problems are together.

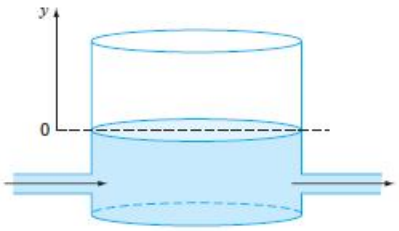


Figure P1.7

1.8 For the same storage tank described in Prob. 1.7, suppose that the outflow is not constant but rather depends on the depth. For this case, the differential equation for depth can be written as

$$\frac{dy}{dx} = 3\frac{Q}{A}\sin^2(t) - \frac{\alpha(1+y)^{1.5}}{A}$$

Use Euler's method to solve for the depth y from $t = 0$ to 10 d with a step size of 0.5 d. The parameter values are $A = 1200$ m^2, $Q = 500$ m^3/d, and $\alpha = 300$. Assume that the initial condition is $y = 0$.

Solution: See code and output below

2.22 A simply supported beam is loaded as shown in Fig. P2.22. Using singularity functions, the displacement along the beam can be expressed by the equation:

$$u_y(x) = \frac{-5}{6}[\langle x-0\rangle^4 - \langle x-5\rangle^4] + \frac{15}{6}\langle x-8\rangle^3 + 75\langle x-7\rangle^2 + \frac{57}{6}x^3 - 238.25x$$

By definition, the singularity function can be expressed as follows:

$$\langle x-a\rangle^n = \begin{cases} (x-a)^n & \text{when } x > a \\ 0 & \text{when } x \le a \end{cases}$$

Develop a program that creates a plot of displacement versus distance along the beam x. Note that $x = 0$ at the left end of the beam.

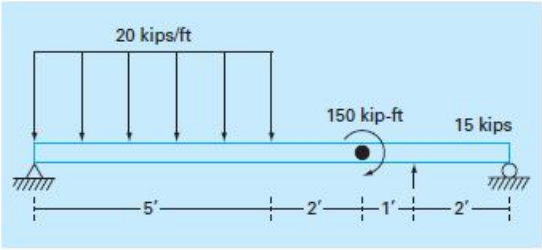
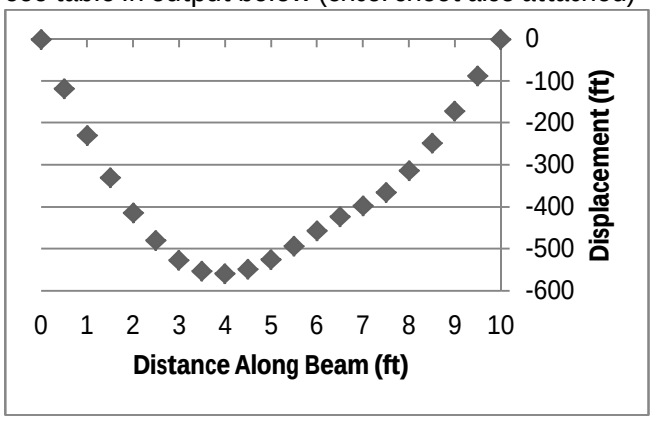


Figure P2.22

Special Instructions:
Write a program using your favorite high-level language (C++, VBA/Excel, VBA/CALC, etc.) to produce a table of values of $u_y(x)$ for $0 < x < 10$ ft, with increment $\Delta x = 0.5$ ft. Using the results from the table produce the required graph by hand or using Excel or CALC.

Solution:
See table in output below (excel sheet also attached)



Special Instructions:
The program must take as input the following values:

- The value whose square root is sought, a
- An initial guess of the solution, x_{guess}
- The number of decimals, n, in order to calculate the error criteria, ε_s, according to equation (3.7)
- The maximum number of iterations allowed before declaring a diverging solution, maxit

The program should produce, as output, the value of the error criteria, ε_s, and a table showing the different iterations required to find a solution using the required error criteria. Show the results of your program for a = 122.5, x_{guess} = 5.0, n = 5, maxit = 20.
Solution: See output below

3.13 The "divide and average" method, an old-time method for approximating the square root of any positive number a, can be formulated as

$$x = \frac{x + a/x}{2}$$

Write a well-structured function to implement this algorithm based on the algorithm outlined in Fig. 3.3.

Output:

```
Problem 1:
ODE - A: 1200
ODE - Q: 500
ODE - alpha: 300
Euler's Method - dt: 0.5
Euler's Method - t_0: 0
Euler's Method - t_f: 10
Euler's Method - y_0: 0

The approximation: 0.546303
--------------------------------------------------

Problem 2:
    x      u(x)
    0         0
  0.5   -117.99
    1  -229.583
  1.5  -329.531
    2  -413.833
  2.5   -479.74
    3   -525.75
  3.5  -551.615
    4  -558.333
  4.5  -548.156
    5  -524.583
  5.5  -492.313
    6  -456.667
  6.5  -423.021
    7   -396.75
  7.5  -364.479
    8  -312.833
  8.5  -246.875
    9  -170.417
  9.5  -86.9583
   10         0
--------------------------------------------------

Problem 3:
a: 122.5
x_guess: 5
maxit: 20
n: 5
    i         x        % error
    1         5
    2     14.75     66.101695%
    3   11.5275     27.954420%
    4   11.0771      4.066121%
    5    11.068      0.082735%
    6    11.068      0.000034%
Press any key to continue . . .
```

main.cpp:

```cpp
#include <iostream>
#include <iomanip>
#include <math.h>
#include "functions.h"
#include "..\shared\methods.cpp"

int main() {
    // Problem 1
    const double
        A = 1200,
        Q = 500,
        alpha = 300,
        Dt = 0.5,
        t_init = 0,
        t_final = 10,
        y_init = 0;

    std::unordered_map<double, double> result = Eulers(
        [A, Q, alpha, Dt, t_init, t_final, y_init](double t, double f) ->
double {
            double sin_t = sin(t);
            return 3 * (Q / A) * sin_t * sin_t - (alpha * pow(1 + f, 1.5)
/ A);
        },
        y_init, t_init, t_final, Dt);

    std::cout
        << "Problem 1: " << std::endl
        << "ODE - A: " << A << std::endl
        << "ODE - Q: " << Q << std::endl
        << "ODE - alpha: " << alpha << std::endl
        << "Euler's Method - dt: " << Dt << std::endl
        << "Euler's Method - t_0: " << t_init << std::endl
        << "Euler's Method - t_f: " << t_final << std::endl
        << "Euler's Method - y_0: " << y_init << std::endl << std:: endl
        << "The approximation: " << result[10] << std::endl
        << "-----------------------------------------------" <<
std::endl << std::endl;


    // Problem 2
    const double
        x_initial = 0,
        x_final = 10,
        x_increment = 0.5;

    std::cout
        << "Problem 2:" << std::endl
        << std::setw(5) << "x" << std::setw(10) << "u(x)" << std::endl;

    for (double x = 0; x <= 10; x += 0.5) {
        std::cout
            << std::setw(5) << x
            << std::setw(10) << displacement(x)
            << std::endl;
    }
```

```
    std::cout
        << "-------------------------------------------------" <<
std::endl << std::endl;

    // Problem 3
    std::streamsize def_precision = std::cout.precision();
    int i = 1, n, max_iterations;
    double x, a, old, error = 100;

    std::cout << "Problem 3:" << std::endl
        << "a: ";
    std::cin >> a;

    std::cout
        << "x_guess: ";
    std::cin >> x;

    std::cout
        << "maxit: ";
    std::cin >> max_iterations;

    std::cout
        << "n: ";
    std::cin >> n;

    double error_criteria = 0.5 * pow(10, 2 - n);

    std::cout << std::endl
        << std::setw(5) << "i" << std::setw(10) << "x" << std::setw(15) <<
"% error" << std::endl
        << std::setw(5) << i  << std::setw(10) << x << std::endl;

    for (i = 2; i < max_iterations && error > error_criteria; ++i) {
        old = x;
        x = (x + (a / x)) / 2;
        error = abs((x - old) / x) * 100;
        std::cout
            << std::setw(5) << i
            << std::setw(10) << x
            << std::setw(14) << std::fixed << std::setprecision(6) <<
error << "%"
            << std::endl << std::defaultfloat <<
std::setprecision(def_precision);
    }

    system("pause");
    return 0;
}
```

functions.h:

```h
#pragma once

#include <math.h>

double displacement(double);
double singularity(double, double, double);
```

functions.cpp:

```cpp
#include "functions.h"

double displacement(double x) {
    return
        (-5 * (singularity(x, 0, 4) - singularity(x, 5, 4)) / 6) +
        (15 * singularity(x, 8, 3) / 6) +
        (75 * singularity(x, 7, 2)) +
        (57 * pow(x, 3) / 6) -
        (238.25 * x);
}

double singularity(double x, double a, double n) {
    if (x <= a) { return 0; }
    return pow(x - a, n);
}
```

methods.h:

```
#pragma once

#include <unordered_map>
#include <functional>

std::unordered_map<double, double> Eulers(std::function<double(double,
double)>, double, double, double, double);
```

methods.cpp:

```
#include "methods.h"

std::unordered_map<double, double> Eulers(
    std::function<double(double, double)> ode,
    double f_init,
    double t_init,
    double t_final,
    double t_inc) {

    double
        t = t_init,
        f = f_init;

    std::unordered_map<double, double> steps;
    steps.insert(std::make_pair(t, f));

    do {
        f += t_inc * ode(t, f);
        t += t_inc;

        steps.insert(std::make_pair(t, f));
    } while (t < t_final);

    return steps;
}
```

4.8 The Stefan-Boltzmann law can be employed to estimate the rate of radiation of energy H from a surface, as in

$$H = Ae\sigma T^4$$

where H is in watts, A = the surface area (m^2), e = the emissivity that characterizes the emitting properties of the surface (dimensionless), σ = a universal constant called the Stefan-Boltzmann constant ($= 5.67 \times 10^{-8}$ W m^{-2} K^{-4}), and T = absolute temperature (K). Determine the error of H for a steel plate with $A = 0.15$ m^2, $e = 0.90$, and $T = 650 \pm 20$. Compare your results with the exact error. Repeat the computation but with $T = 650 \pm 40$. Interpret your results.

Solution:

$$H'(T) = 4ae\sigma T^3 \implies \Delta H = H'(T)\Delta T = 8.40847 * 40 = 336.339$$

4.12 Evaluate and interpret the condition numbers for

(a) $f(x) = \sqrt{|x-1|+1}$ for $x = 1.00001$

(b) $f(x) = e^{-x}$ for $x = 10$

(c) $f(x) = \sqrt{x^2+1} - x$ for $x = 300$

(d) $f(x) = \dfrac{e^{-x}-1}{x}$ for $x = 0.001$

(e) $f(x) = \dfrac{\sin x}{1+\cos x}$ for $x = 1.0001\pi$

Special instructions: just do part e

Solution:
I punched it into my calculator, and I get: -10,000.86