Chapter 1 – Problem 1.8

A storage tank contains a liquid at depth y where $y = 0$ when the tank is half full. The equation for the depth can be written as

$$\begin{aligned} \frac{dy}{dt} &= 3\frac{Q}{A}\sin^2(t) - \frac{\alpha(1+y)^{1.5}}{A} \\ \begin{pmatrix}\text{change in}\\ \text{volume}\end{pmatrix} &= (\text{inflow}) - (\text{outflow}) \end{aligned}$$

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 $A = 1200$ m², $Q = 500$ m³/d, and $\alpha = 300$. Assume that the initial condition is $y = 0$.

```cpp
// 01/21/2014 - ENGR 2450 - Meine, Joel
// Chapter 1 - Problem 1.8

#include <iostream>
#include <math.h>
using namespace std;

int main()
{
	double A = 1200; // Area (m^2)
	double Q = 500; // Flow (m^3/d)
	double a = 300; // Constant
	double y = 0; // Depth_initial (m)

	std::cout << "Chapter 1 - Problem 1.8" << std::endl;
	std::cout << "============================" << std::endl;
	std::cout << "Area, A(m^2) = " << A << std::endl;
	std::cout << "Flow, Q(m^3/d) = " << Q << std::endl;
	std::cout << "Constant, a = " << a << std::endl;
	std::cout << "Depth_initial, y(m) = " << y << std::endl;
	std::cout << "----------------------------" << std::endl;
	std::cout << " t(s)     y(m)              " << std::endl;
	std::cout << "----------------------------" << std::endl;

	double  t_f = 10,  t_s = 0.5;
	int t = 0, T = t_f / t_s;

	printf(" %2.1f %12.10f \n", t, y);
	for (t; t < T; t++)
	{
		// value_new = value_old + step_size*slope
		y = y + t_s*(3*(Q/A)*pow(sin(t*t_s),2) - (a*pow((1+y),1.5))/A);
		printf(" %2.1f %12.10f \n", t*t_s + t_s, y);
	}
	std::cout << "----------------------------" << std::endl;
	system("pause");
	return 0;
}
```

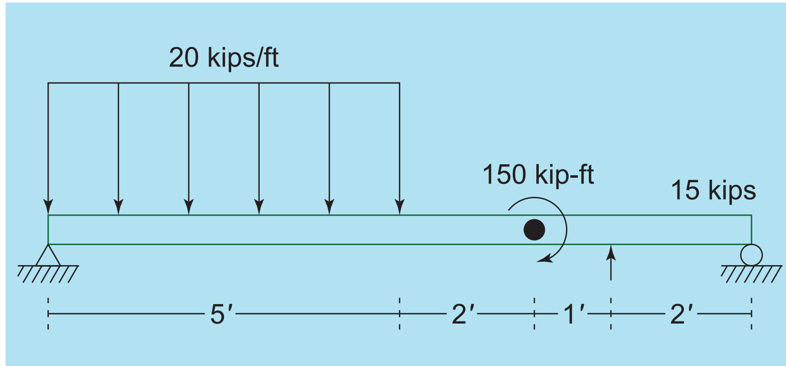
```
Chapter 1 - Problem 1.8
============================
Area, A(m^2) = 1200
Flow, Q(m^3/d) = 500
Constant, a = 300
Depth_initial, y(m) = 0
----------------------------
 t(s)     y(m)
----------------------------
 0.0 0.0000000000
 0.5 -0.1250000000
 1.0 -0.0836554148
 1.5 0.2492430964
 2.0 0.6965815876
 2.5 0.9371143962
 3.0 0.8239598637
 3.5 0.5284901721
 4.0 0.3691826123
 4.5 0.5268877466
 5.0 0.8882742094
 5.5 1.1386387651
 6.0 1.0588102828
 6.5 0.7383440155
 7.0 0.4807741680
 7.5 0.5253049601
 8.0 0.8397323075
 8.5 1.1395817976
 9.0 1.1468662795
 9.5 0.8598136904
 10.0 0.5463034139
----------------------------
Press any key to continue . . .
```

Chapter 2 – Problem 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:

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

Figure P2.22



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

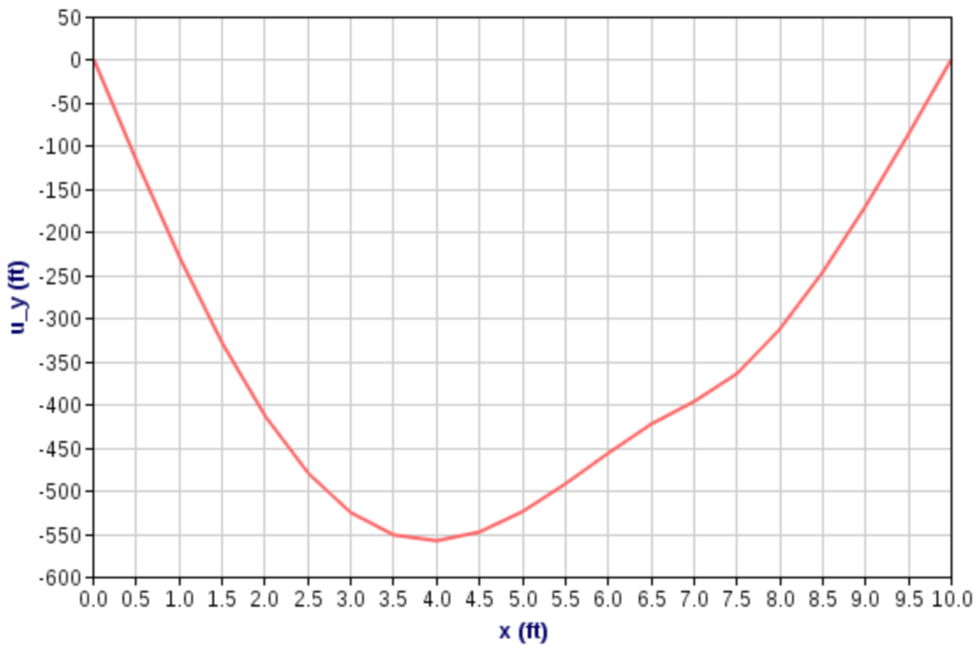
$$\langle x-a\rangle^n = \begin{cases} (x-a)^n & \text{when } x > a \\ 0 & \text{when } x \le a \end{cases}$$

Develop a program that crates a plot of displacement verses distance along the beam x. Note that x = 0 at the left end of the beam.

```
// 01/21/2014 - ENGR 2450 - Meine, Joel
// Chapter 2 - Problem 2.22

#include <iostream>
#include <math.h>
using namespace std;

int main()
{
	double x_i = 0; // Position_initial (ft)
	double x_f = 10; // Position_final (ft)
	double dx = 0.5; // Position_step (ft)
	double x = 0; // Position_current (ft)

	std::cout << "Chapter 2 - Problem 2.22" << std::endl;
	std::cout << "============================" << std::endl;
	std::cout << "Position_initial, x_i(ft) = " << x_i << 
std::endl;
	std::cout << "Position_final, x_f(ft) = " << x_f << std::endl;
	std::cout << "Position_step, dx(ft) = " << dx << std::endl;
	std::cout << "Position_current, x(ft)" << std::endl;
	std::cout << "Beam Displacement, u_y(ft)" << std::endl;
	std::cout << "----------------------------" << std::endl;
	std::cout << "x(ft)   u_y(ft)             " << std::endl;
	std::cout << "----------------------------" << std::endl;

	int n = (x_f-x_i)/dx + 1;
	for (int i = 1; i <= n; i++)
	{
		x = x_i + (i - 1)*dx;
		double u_y = 0;
		if (x <= 0)
			u_y = (57.0/6.0)*pow(x,3) - 238.25*x;
```

```
Chapter 2 - Problem 2.22
============================
Position_initial, x_i(ft) = 0
Position_final, x_f(ft) = 10
Position_step, dx(ft) = 0.5
Position_current, x(ft)
Beam Displacement, u_y(ft)
----------------------------
x(ft)   u_y(ft)
----------------------------
 0.0 0.0000000000
 0.5 -117.9895833333
 1.0 -229.5833333333
 1.5 -329.5312500000
 2.0 -413.8333333333
 2.5 -479.7395833333
 3.0 -525.7500000000
 3.5 -551.6145833333
 4.0 -558.3333333333
 4.5 -548.1562500000
 5.0 -524.5833333333
 5.5 -492.3125000000
 6.0 -456.6666666667
 6.5 -423.0208333333
 7.0 -396.7500000000
 7.5 -364.4791666667
 8.0 -312.8333333333
 8.5 -246.8750000000
 9.0 -170.4166666667
 9.5 -86.9583333333
 10.0 0.0000000000
----------------------------
Press any key to continue . . .
```

```
            else if (x > 0 && x <= 5.0)
                    u_y = (-5.0/6.0)*pow(x, 4) + (57.0/6.0)*pow(x,3) - 238.25*x;
            else if (x > 5.0 && x <= 7.0)
                    u_y = (-5.0/6.0)*(pow(x,4)-pow(x-5,4)) + (57.0/6.0)*pow(x,3) - 238.25*x;
            else if (x > 7.0 && x <= 8.0)
                    u_y = (-5.0/6.0)*(pow(x,4)-pow(x-5,4)) + 75.0*pow(x-7,2) + (57.0/6.0)*pow(x,3) -
                          238.25*x;
            else if (x > 8.0)
                    u_y = (-5.0/6.0)*(pow(x,4) - pow(x-5,4)) + (15.0/6.0)*pow(x-8,3) + 75.0*pow(x-7,2)
                          + (57.0/6.0)*pow(x,3) - 238.25*x;
            else
                    std::cout << "ERROR!" << std::endl;
            printf(" %2.1f %12.10f \n", x, u_y);
        }
        std::cout << "----------------------------" << std::endl;
        system("pause");
        return 0;
}
```



Chapter 3 – Problem 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 as follows

```
FUNCTION IterMeth(val, es, maxit)
iter = 1
sol = val
ea = 100
DO
   solold = sol
   sol = ...
   iter = iter + 1
   IF sol ≠ 0 ea=abs((sol − solold)/sol)*100
   IF ea ≤ es OR iter ≥ maxit EXIT
END DO
IterMeth = sol
END IterMeth
```

$$\varepsilon_a = \left(\frac{\text{current approximation} - \text{previous approximation}}{\text{current approximation}} \times 100 \right)\% \quad (3.5)$$

$$\varepsilon_S = \left(0.5 \times 10^{2-n}\right)\% \quad (3.7)$$

The algorithm 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 algorithm 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. The table, thus, should show:

Iteration i	Current value of x	% rel. error ε_a
1	x_{guess}	–
2	x_2	ε_{a2}
3	…	…
4		
…		

Show the results of your program $a = 122.5$, $x_{guess} = 5.0$, $n = 5$, $maxit = 20$.

```
// 01/21/2014 - ENGR 2450 - Meine, Joel
// Chapter 3 - Problem 3.13

#include <iostream>
#include <math.h>
using namespace std;

int main()
{
	double x_guess = 5.0; // Guess_initial
	int n = 5; // Decimals_number
	double es = 0.5*pow(10,2-n); // Error Criteria
	int maxit = 20; // Maximum Number of Iterations
	double sol; // Solution_current
	double solold; // Solution_previous
	int iter; // Iteration
	double a = 122.5; // Square Root Input
	double ea; // Relative Error

	std::cout << "Chapter 3 - Problem 3.13" << std::endl;
	std::cout << "============================" <<
std::endl;
	std::cout << "Guess_initial, x_guess = " << x_guess << std::endl;
	std::cout << "Decimals_number, n = " << n << std::endl;
	std::cout << "Error Criteria, es = " << es << std::endl;
	std::cout << "Maximum Number of Iterations, maxit = " << maxit << std::endl;
	std::cout << "Square Root Input, a = " << a << std::endl;
	std::cout << "----------------------------" << std::endl;
	std::cout << "iter.  x_current  rel. error" << std::endl;
	std::cout << "----------------------------" << std::endl;

	iter = 1; sol = x_guess; ea = 100;
	printf(" %2i %12.10f %12.10f ", iter, sol, ea);
	do {
		solold = sol;
		sol = 0.5*(sol+(a/sol)); // Divide-and-Average Method
		iter = iter + 1;
		if (sol != 0) ea = abs((sol-solold)/sol)*100;
		printf("\n %2i %12.10f %12.10f ", iter, sol, ea);
	} while (ea > es && iter < maxit);
	if (ea <= es)
	{
		std::cout << "\n----------------------------" << std::endl;
		printf("x_solution = %12.10f \n", sol);
		printf("\nrel. error = %12.10f \n", ea);
		std::cout << "----------------------------" << std::endl;
	}
	else if (iter >= maxit)
		std::cout << "No Solution due to Divergence" << std::endl;
	system("pause");
	return 0;
}
```

```
Chapter 3 - Problem 3.13
============================
Guess_initial, x_guess = 5
Decimals_number, n = 5
Error Criteria, es = 0.0005
Maximum Number of Iterations, maxit = 20
Square Root Input, a = 122.5
----------------------------
iter.  x_current  rel. error
----------------------------
  1 5.0000000000 100.0000000000
  2 14.7500000000 66.1016949153
  3 11.5275423729 27.9544201434
  4 11.0771327009 4.0661214791
  5 11.0679755987 0.0827351138
  6 11.0679718106 0.0000342255
----------------------------
x_solution = 11.0679718106

rel. error = 0.0000342255
----------------------------
Press any key to continue . . .
```

Chapter 4 – Problem 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 (m²), *e* = the emissivity that characterizes the emitting properties of the surface (dimensionless), *σ* = a universal constant called the Stefan-Boltzmann constant (= 5.67×10^{-8} W m^{-2} K^{-4}), and *T* = absolute temperature (K). Determine the error of *H* for a steel plate with *A* = 0.15 m², *e* = 0.90, and *T* = 650 ± 20. Compare your results with the exact error. Repeat the computation but with *T* = 650 ± 40. Interpret your results.

Interpretation

The calculated values of the error approximation and exact error yield reasonably the same results with marginal difference. However, we see that as the variance value associated with the independent variable ΔT increases, the difference between the error approximation and the exact error likewise increases. Therefore it is expected that as the variance value associated with a given independent variable increases, the error approximation will be less true to the exact error.

Error Approximation $= \Delta H_A(\tilde{T}) = |H'(\tilde{T})|\Delta\tilde{T}$

Exact Error $= \Delta H_E(\tilde{T}, \Delta\tilde{T}) = 0.5|H(\tilde{T}-\Delta\tilde{T}) - H(\tilde{T}+\Delta\tilde{T})|$

$H(T) = Ae\sigma T^4$ W $\quad A = 0.15$ m^2 $\quad e = 0.90$

$H'(T) = \frac{dH}{dT} = 4Ae\sigma T^3$ W $\quad \sigma = 5.67 \times 10^{-8}$ W m^{-2} K^{-4}

$\tilde{T} = 650$ K

$\Delta\tilde{T}_1 = 20$ K $\quad \Delta\tilde{T}_2 = 40$ K

$\Rightarrow H(\tilde{T}-\Delta\tilde{T}_1) = 1205.81$ W $\quad H(\tilde{T}+\Delta\tilde{T}_1) = 1542.47$ W

$\therefore \Delta H_{E_1}(\tilde{T}, \Delta\tilde{T}_1) = 168.3285786$ W

$\Rightarrow H(\tilde{T}-\Delta\tilde{T}_2) = 1059.83$ W $\quad H(\tilde{T}+\Delta\tilde{T}_2) = 1735.05$ W

$\therefore \Delta H_{E_2}(\tilde{T}, \Delta\tilde{T}_2) = 337.6124388$ W

$\Rightarrow H'(\tilde{T}) = 8.40846825$ W/K

$\therefore \Delta H_{A_1}(\tilde{T}, \Delta\tilde{T}_1) = 168.169365$ W

$\therefore \Delta H_{A_2}(\tilde{T}, \Delta\tilde{T}_2) = 336.33873$ W

$\Rightarrow |\Delta H_{E_1} - \Delta H_{A_1}| = 0.1592136$

$|\Delta H_{E_2} - \Delta H_{A_2}| = 1.2737088$

Chapter 4 – Problem 4.12(e)

Evaluate and interpret the condition numbers for

$$f(x) = \frac{\sin x}{1+\cos x} \qquad \text{for } x = 1.0001\pi$$

Interpretation

The condition number is less than 1 and therefore the relative error in $f(x)$ is attenuated.

$$\text{Condition Number} = \frac{\tilde{x} f'(\tilde{x})}{f(\tilde{x})}$$

$$\tilde{x} = 1.0001\pi \qquad f(x) = \frac{\sin x}{1+\cos x}$$

$$f'(x) = \frac{1}{1+\cos x}$$

$$f(\tilde{x}) = -6366.198$$

$$f'(\tilde{x}) = 20264237.552$$

$$\therefore \text{Condition Number} = \underline{\underline{-10001.000163204}}$$