1. Define the smallest positive root of the polynomial $x^3 + 3.8x^2 - 8.6x - 24.4 = 0$.

(a) Use Newton-Raphson's method to determine the solution in the first five iterations. Use as a first estimate x=2.
(b) Plot the function and select two points near the root to start the solution process with the secant method by using five iterations.

Use four decimal points and compare the results. Which method converges faster? Solve the problem **by hand**. (25%)

2. The centroid *c* of a circular sector is provided by the equation: $x = \frac{2r\sin\theta}{3\theta}$. Find the angle θ so that $x = \frac{r}{2}$, by using the bisection method with starting points a=1 and b=2. Stop the process when the $\text{tol}_i = \frac{|b_i - a_i|}{2} \le 0.002$, where *i* is the number of iterations. In every step, calculate also the Estimated Relative Error $\text{ERE}_i = \left|\frac{x_{NS}^{(i)} - x_{NS}^{(i-1)}}{x_{NS}^{(i-1)}}\right|$, where $x_{NS}^{(i)}$ is the numerical solution at i^{th} iteration. Use eight decimal points. Solve the problem **by hand**. (25%)

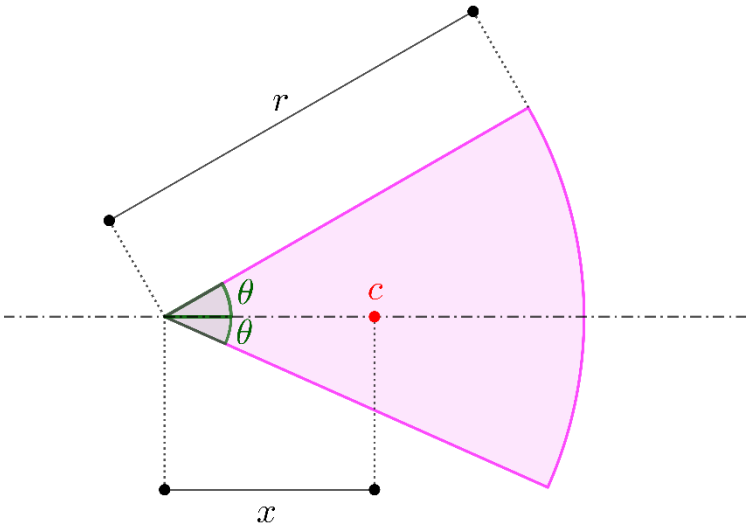

Fig. 1. Centroid c of circular sector

3. Write a user-defined function in MatLab named `RegulaFalsiRootMod1` that solves the nonlinear equation f(x)=0 with modified Regula Falsi method (see Fig.2). The function should check if the endpoints stay the same. If an endpoint stays the same for three iterations, the value of the function at that point should be divided by 2. If the value of the point stays the same for three more iterations the value of the function at that point should be divided again by 2 (the true value is now divided by 4), and so on. The function should have as input arguments, the function (function handle), two values that bracket the root and the error. The function should stop when the Estimated

Relative Error $\text{ERE}_i = \left| \frac{x_{NS}^{(i)} - x_{NS}^{(i-1)}}{x_{NS}^{(i-1)}} \right|$ becomes smaller or equal to the error (inserted as an input argument). If the input arguments are less than 4, the function should automatically define the error equal to 0.0001. The maximum number of iterations should be imax=100 and when it has been reached the program should print a message that states that the solution has not been derived in imax iterations should be shown.

Use the function `RegulaFalsiRootMod1` you wrote, to find the smallest positive root of the polynomial $x^3 + 12x^2 - 100x - 6 = 0$ with the initial two values to be 4 and 6. Compare the result you obtained by using MatLab build-in functions `fzero` and `roots`. Use format long to compare as many figures as possible.

For the `RegulaFalsiRootMod1` function you are allowed to use the following build-in functions of MatLab: feval, nargin, error, rem, abs, fprintf. Submit a MatLab files with explanations and report with comparisons and explanations. (25%)

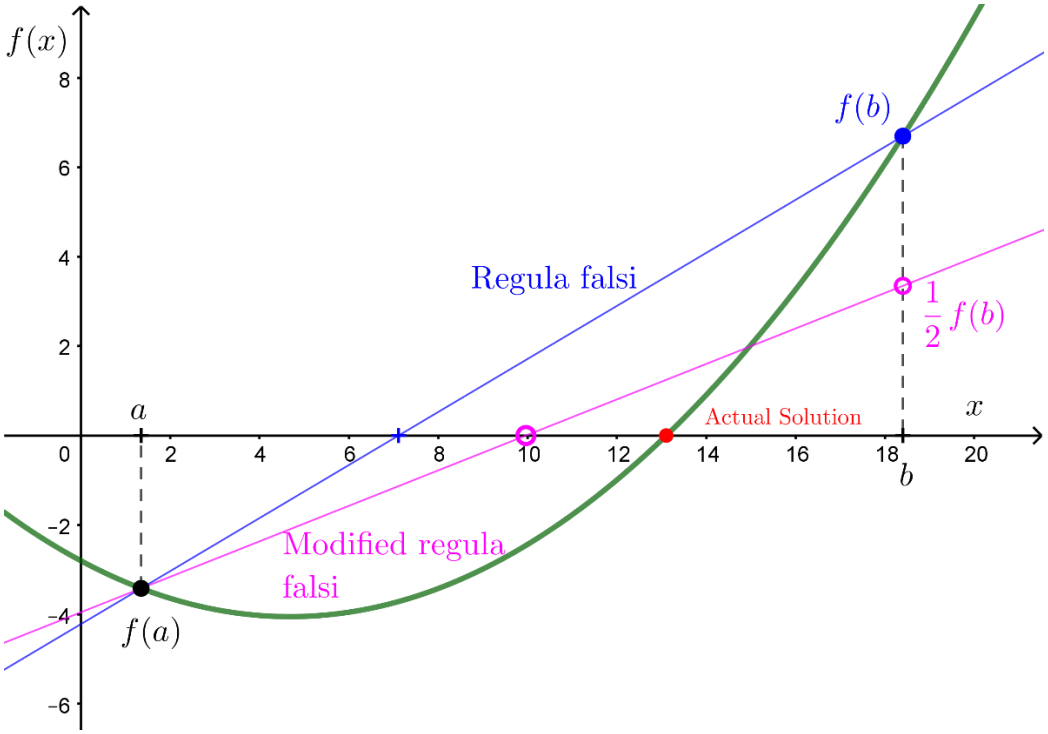

Fig. 2. Modified Regula falsi method (modification 1, see lecture slides)

4. Write a user-defined function `NewtonManyVariables(x,f,jf,tol)` where `x` is the initial approximation in column array, `f` is a column array containing the functions, `jf` is the Jacobian matrix and `tol` is the tolerance. The function should be able to solve a system of nonlinear equations by using the Newton's method for a system of nonlinear equations. The user-defined function should return the roots and number of iterations. The functions `f` and the matrix `jf` should be given with function handle (@). The function should stop when the tolerance has been achieved. Use the Euclidean norm (norm in MatLab) of the function as tolerance measure. If the tolerance

is not given when the function is called, define it equal to 0.00005. After developing the function use it solve the following system of equations and report the roots and the number of iterations, (use as initial values x=5 and y=1 with tolerance 0.00005):

$$2x^2y = 1$$
$$\frac{x^3}{20} + y = 10$$

For this assignment you can only use the following build-in functions of MatLab feval, \, norm, nargin, Use the following set of equations and not the Cramer'r rule: (25%)

System of nonlinear equations - Newton - General case

- In the general case of any number of variables and equations, we can write the system of equations as:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

where $\mathbf{f}$ and $\mathbf{x}$ are column vectors of n components. For the rth iteration (using only the first two terms):

$$\mathbf{x}^{r+1} = \mathbf{x}^r + \Delta\mathbf{x}^r; \quad r = 0, 1, 2, 3...$$

- Using the Taylor series expansion in n-dimension:

$$\mathbf{f}(\mathbf{x}^r + \Delta\mathbf{x}^r) = \mathbf{f}(\mathbf{x}^r) + \mathbf{J}_r\Delta\mathbf{x}^r$$

where $\mathbf{J}_r = [\partial f_i(\mathbf{x}^r)/\partial x_j]$ for $i = 1, 2, .., n$ and $j = 1, 2, .., n$ is the Jacobian matrix (which may be singular and the inverse cannot be calculated).

- Finally, solving for the improved approximation by assuming that $\mathbf{f}(\mathbf{x}^r + \Delta\mathbf{x}^r) = \mathbf{f}(\mathbf{x}^r) + \mathbf{J}_r\Delta\mathbf{x}^r \approx 0$:

$$\mathbf{x}^{r+1} = \mathbf{x}^r - \mathbf{J}_r^{-1}\mathbf{f}(\mathbf{x}^r); \quad r = 0, 1, 2, 3, ...$$

- Limitations of method: (1) needs good initial approximation and (2) provide derivatives with respect each variable.

Grading criteria:
Correctness
Justification
Efficiency
Presentation