1. ค่าคลาดเคลื่อนสัมบูรณ์ (E_{abs}) :
$$E_{abs} = |\text{ ค่าจริง } - \text{ ค่าประมาณ }|$$

2. ค่าคลาดเคลื่อนสัมพัทธ์ (E_{rel}):
$$E_{rel} = \frac{|\text{ ค่าจริง } - \text{ ค่าประมาณ }|}{|\text{ ค่าจริง }|}$$

3. ร้อยละของค่าความคลาดเคลื่อนสัมพัทธ์ (ε_t) :
$$\varepsilon_t = \frac{|\text{ ค่าจริง } - \text{ ค่าประมาณ }|}{|\text{ ค่าจริง }|} \times 100\%$$

4. ร้อยละของค่าคลาดเคลื่อนเปรียบเทียบกับค่าประมาณ (ε_a) :
$$\varepsilon_a = \frac{|\text{ ค่าประมาณสุดท้าย } - \text{ ค่าประมาณก่อนสุดท้าย }|}{|\text{ ค่าประมาณสุดท้าย }|} \times 100\%$$

5. ร้อยละของค่าความคลาดเคลื่อนสัมพัทธ์ (ε_t) :
$$\varepsilon_t = \frac{|\text{ ค่าจริง } - \text{ ค่าประมาณ }|}{|\text{ ค่าจริง }|} \times 100\%$$

6. ร้อยละของค่าคลาดเคลื่อนเปรียบเทียบกับค่าประมาณ (ε_a) :
$$\varepsilon_a = \frac{|\text{ ค่าประมาณสุดท้าย } - \text{ ค่าประมาณก่อนสุดท้าย }|}{|\text{ ค่าประมาณสุดท้าย }|} \times 100\%$$

7. Bisection method :
$$x_r = \frac{x_l + x_u}{2}$$

8. False position method :
$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

9. Fixed point Iteration Method
$$x_{i+1} = g(x_i)$$

10. Newton Raphson Method :
$$x_{i+1} = x_i - \frac{f(x_i)}{f\,'(x_i)}$$

11. Secant Method:
$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

12. Modified Newton-Raphson Method :
$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)}$$

13. Cramer's Rule :
$$x_i = \frac{\det(A_i)}{\det(A)}$$

14. Matrix Inversion :

$$AX = B \iff X = A^{-1}B$$

15. Scoop Installation (via Windows PowerShell):

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
Invoke-RestMethod -Uri https://get.scoop.sh | Invoke-Expression
```

16. Python Installation (via Scoop):

```
scoop install python
```

17. Python Versions

```
python --version
```

18. Installation of Python Packages

```
pip install jupyter numpy matplotlib
```

19. Running Jupyter

```
jupyter lab
```

```
python -m notebook
```

20. Defined Numpy Function for Matrix Row Swap:

```python
def RowSwap(A, k, l):
# ============================================================================
#     A is a NumPy array.  RowSwap will return duplicate array with rows
#     k and l swapped.
# ============================================================================
    m = A.shape[0]  # m is number of rows in A
    n = A.shape[1]  # n is number of columns in A

    B = np.copy(A).astype('float64')

    for j in range(n):
        temp = B[k][j]
        B[k][j] = B[l][j]
        B[l][j] = temp

    return B
```

21. Defined Numpy Function for Matrix Row Scale:

```
def RowScale(A, k, scale):
# ================================================================================
#     A is a NumPy array.  RowScale will return duplicate array with the
#     entries of row k multiplied by scale.
# ================================================================================
    m = A.shape[0]  # m is number of rows in A
    n = A.shape[1]  # n is number of columns in A

    B = np.copy(A).astype('float64')

    for j in range(n):
        B[k][j] *= scale

    return B
```

22. Defined Numpy Function for Matrix Row Add:

```
def RowAdd(A, k, l, scale):
# ================================================================================
#     A is a numpy array.  RowAdd will return duplicate array with row
#     l modifed.  The new values will be the old values of row l added to
#     the values of row k, multiplied by scale.
# ================================================================================
    m = A.shape[0]  # m is number of rows in A
    n = A.shape[1]  # n is number of columns in A

    B = np.copy(A).astype('float64')

    for j in range(n):
        B[l][j] += B[k][j]*scale

    return B
```