

# Practice Exam 2 – Numerical Algebra

Ch En 263 – Numerical Tools

## Instructions

- You have 75 minutes to complete the exam.
- You **may** use your notes, the internet, help menus, etc. (i.e. the exam is “open book”). You may not look at another persons exam or ask them for help. You may of course ask clarifying questions to Dr. Tree or the TA.
- You need a computer to complete this exam. You may use either the lab computer or your own computer, but you may not use a handheld calculator. You may also use scratch paper or write on your test, but neither will be accepted for credit.
- A **required** Excel template and necessary data files are provided on Learning Suite. **Enter your answers in the gray colored cells in the Excel workbook.** Also, please do not move the colored cells in the Excel workbook.
- Submit your exam to Learning Suite **at the end of class**. You will submit three files.
  - Lastname\_Firstname\_Exam2.xlsx
  - Lastname\_Firstname\_Exam2.py

## Contents

This exam contains:

- 12 Qualitative Questions (3 pts each, 39 pts)
- 6 Quantitative Multiple Choice Questions (10 pts each, 60 pts)

## I. Qualitative Questions (39 pts)

Answer the indicated question with either True (T) or False (F), the multiple choice letter or a short answer as indicated. Enter your answer into the “Multiple\_Choice” worksheet in the Excel workbook named “Lastname\_Firstname\_Exam2.xlsx.” No partial credit will be given.

- \_\_\_\_\_ (True or False) In root finding methods, one tries to find where the square of the residual is a minimum, but in optimization methods one tries to find where the residual crosses zero.
- \_\_\_\_\_ (True or False) Picard’s method is preferred for solving nonlinear equations, because it is both easier to write and more reliable than Newton’s method.
- \_\_\_\_\_ (True or False) In an iterative method, we converge to a solution when  $|x^{(k+1)} - x^{(k)}| \rightarrow 0$  regardless of the value of the residual.
- \_\_\_\_\_ (True or False) The “cost” of a computation includes the memory that variables consume and calculations that consume the CPU time.
- \_\_\_\_\_ (True or False) When solving nonlinear “Engineering” equations, it is not necessary to put them in residual form because of the units (but it is still a good idea).
- The system of linear equations

$$\begin{aligned} 5x - y + 3z &= 7 \\ x + 3y - 2z &= 1 \\ -2x - 2y + 5z &= -3 \end{aligned}$$

can be re-written in the form  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ , where  $\mathbf{A}$  is a matrix, and  $\mathbf{x}$  and  $\mathbf{b}$  are vectors. Which are the correct  $\mathbf{A}$  and  $\mathbf{b}$  for the system shown above.

$$\begin{aligned} \text{(a) } \mathbf{A} &= \begin{bmatrix} 5 & 1 & -2 \\ -1 & 3 & -2 \\ 3 & -2 & 5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -7 \\ -1 \\ 3 \end{bmatrix} & \text{(b) } \mathbf{A} &= \begin{bmatrix} 5 & 1 & -2 \\ -1 & 3 & -2 \\ 3 & -2 & 5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 7 \\ 1 \\ -3 \end{bmatrix} \\ \text{(c) } \mathbf{A} &= \begin{bmatrix} 5 & -1 & 3 \\ 1 & 3 & -2 \\ -2 & -2 & 5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -7 \\ -1 \\ 3 \end{bmatrix} & \text{(d) } \mathbf{A} &= \begin{bmatrix} 5 & -1 & 3 \\ 1 & 3 & -2 \\ -2 & -2 & 5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 7 \\ 1 \\ -3 \end{bmatrix} \end{aligned}$$

- Which of the following is the correct formula for Newton’s method for a single nonlinear equation:
 

(a)  $x^{(k+1)} = x^{(k)} - f(x^{(k)})/f'(x^{(k)})$

(c)  $x_i = \frac{1}{a_{ii}} \left( b_i - \sum_{j=i+1}^{n-1} a_{ij} x_j \right)$

(b)  $x^{(k+1)} = x^{(k)} + f(x^{(k)})$

(d)  $x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{\substack{j=0 \\ j \neq i}}^{n-1} a_{ij} x_j^{(k)} \right)$
- Which method would be appropriate for solving this system of equations:

$$\begin{aligned} x &= 3y - \sin(7\pi/2) \\ 9x - y &= 12 \end{aligned}$$

- (a) A fixed-point method                      (b) Picard's method  
 (c) Newton's method                            (d) Gauss Elimination

9. The Python function `outer_prod` takes two size- $n$  arrays `a` and `b` as arguments and returns a matrix `c`. What is the asymptotic running time,  $T(n)$  of this function for large  $n$ ?

```
def outer_prod(a, b):
    c = np.zeros((n,n))
    for i in range(n):
        for j in range(n):
            c[i, j] = a[i]*b[j]
    return c
```

- (a)  $T(n) = O(1)$             (b)  $T(n) = O(n)$             (c)  $T(n) = O(n^2)$             (d)  $T(n) = O(n^3)$
10. What are ways to come up with a good guess when solving a nonlinear equation? Select all that apply.
- (a) A plot  
 (b) By taking the derivative  
 (c) Mathematical or physical bounds  
 (d) Physical intuition
11. A system of *nonlinear* equations where the number of equations is equal to the number of unknowns is guaranteed to have
- (a) No solution  
 (b) One solution  
 (c) Multiple solutions  
 (d) There is no guarantee in general
12. Picards method is
- (a) a simple, but sometimes unreliable method for solving a linear equation  
 (b) a simple, but sometimes unreliable method for solving a nonlinear equation  
 (c) a method for solving a linear equation that requires one to compute the derivative  
 (d) a method for solving a nonlinear equation that requires one to compute the derivative
13. Choose the answer that correctly writes the system of equations in standard/residual form

$$a^2 + 3b^2 = 4$$

$$a^2 + c^2 = 1$$

$$2b^2 + c^2 = 7$$

where  $x = [a, b, c]^T$ .

$$(a) \quad f(x) = \begin{bmatrix} x_0^2 + 3x_1^2 - 4 \\ x_0^2 + x_2^2 - 1 \\ 2x_1^2 + x_2^2 - 7 \end{bmatrix}$$

$$(b) \quad f(x) = \begin{bmatrix} x_0^2 + 3x_1^2 - 4 \\ x_0^2 + x_1^2 - 1 \\ 2x_0^2 + x_1^2 - 7 \end{bmatrix}$$

$$(c) \quad f(x) = \begin{bmatrix} x_0^2 + 3x_1^2 \\ x_0^2 + x_2^2 \\ 2x_1^2 + x_2^2 \end{bmatrix}$$

$$(d) \quad f(x) = \begin{bmatrix} x_0^2 + 3x_1^2 \\ x_0^2 + x_1^2 \\ 2x_0^2 + x_1^2 \end{bmatrix}$$

## II. Quantitative Questions (60 pts)

Enter your answer into the “Multiple\_Choice” worksheet in the Excel workbook named **Last-name\_Firstname\_Exam2.xlsx**. Show your work for Problems 14-15 in the Excel Workbook and for Problems 16-19 in a Python file named **Lastname\_Firstname\_Exam2\_MC.py**. *You need to show your work to get credit for these problems.*

14. Use Newton’s method in Excel to find a value of  $t$  which satisfies this expression:

$$t^{1/2} - t = -\frac{1}{2}$$

15. The Van der Waals equation gives a relationship between the pressure, molar volume and temperature of a pure component fluid in either the gaseous or liquid state. A dimensionless version of this equation is given by

$$P_R = \frac{\frac{8}{3}T_R}{V_R - \frac{1}{3}} - \frac{3}{V_R^2}$$

where  $P_R$  is a dimensionless pressure,  $V_R$  is a dimensionless molar volume and  $T_R$  is a dimensionless temperature. Use the method of your choice in Excel to find at least one of the three molar volumes  $V_R$  that satisfy this equation when  $T_R = 0.88$  and  $P_R = 0.55$ .

*Hints: (i) There are no units in this problem. (ii) There is an asymptote at  $V_R = 1/3$ . The physically realistic values occur when  $V_R > 1/3$ .*

16. Calculate the norm of the vector in Python.

$$[-5, 3, -2, 0, 4, -3, 1]$$

17. Use Python to solve the system of equations. Report the value of  $z$ .

$$\begin{aligned} 5w + 7x - y + z &= -1 \\ 2w + 6x - 5y + 6z &= 18 \\ -w + 5x + 8y + 6z &= -8 \\ 2w - 5x - 8y - 4z &= 4 \end{aligned}$$

18. Use the method of your choosing to find a solution for  $s$  in Python.

$$s^5 + 3s^2 = 5$$

19. The code below (which has been typed for you in the template file) is a buggy implementation of one of the methods we learned for solving linear systems. Identify the method in the comments in your code and debug the code. Use the debugged code to solve  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$  where

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 2 \\ 0 & 4 & -4 \\ 3 & 1 & -5 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} 5.09 \\ -1.07 \\ -4.03 \end{bmatrix}.$$

Enter  $x[0]$  in the answer sheet.

```
def Solve(A, b):

    n = len(b)

    for k in range(0, n-1):
        for i in range(k+1, n):
            ratio = A[i,i]/A[k,k]
            b[i] -= ratio*b[k]
            for j in range(k, n):
                A[i,j] -= ratio*A[k,j]

    x = np.zeros(n)
    x[n-1] = b[n-1]/A[n-1,n-1]
    for i in range(n-2, -1, -1):
        xi_sum = 0
        for j in range(i+1, n):
            xi_sum = A[i,j]*x[j]

        x[i] = (b[i] - xi_sum)/A[i,i]

    return x
```