National University of Computer and Emerging Sciences
Islamabad Campus

Numerical Computing (CS2008)

Sessional-II Exam

Total Time (Hrs): 1
Total Marks: 75
Total Questions: 6

Course Instructor(s):
Dr.-Ing. Mukhtar Ullah, Dr.Ir. Imran Ashraf,
Dr. Mir Suleman Sarwar, Almas Khan

Date: Nov 2, 2024

_____________ _____________ _________________________
Roll No **Course Section** **Student Signature**

Instructions
Wherever calculations are required, clearly show the formula you used and other necessary steps.
2 bonus points to solve each question as well its parts in order.
Do not write below this line.

Attempt all the questions.

[CLO 6: Approximation of root]

Q1: **[2 + 3*2 + 5 + 2 = 15 Marks]**

Consider the equation $f(x)=x^3-4x+1=0$, and suppose we want to find a root of this equation using the Fixed-Point Iteration Method. Two possible choices of g(x) are given below:

$g_1(x) = \frac{x^3+1}{4}$

$g_2(x) = \frac{x-(x^3-4x+1)}{5}$

a. Rewrite the equation f(x)=0 in the form x=g(x) by proposing one more possible choice for g(x). Name your proposal as $g_3(x)$.

Ans.
Another form of g(x) is given as below.
$g_3(x) = \sqrt[3]{4x-1}$

b. For each g(x), that is $g_1(x)$, $g_2(x)$ and $g_3(x)$, determine if the fixed-point iteration method will converge? Choose an initial guess $x_0 = 1$. Clearly show the required steps to support your answer.

Ans.
For convergence check, we need the derivatives of each g(x) and at $x_0 = 1$. If $|g'(x)| < 1$ near the root, the iteration should converge.

1. $g_1(x) = (x^3 + 1) / 4$; $g_1'(x) = 3*(x**2)/4$, and at x = 1, $g1'(x) \approx 3/4$ (< 1, likely to converge).

2. Given $g_2(x)$ the $g_2'(x) = 1/5\ (1-3x^2-4)$ at $x=1 \approx -6/5$

3. $g_3(x) = \sqrt[3]{4x-1}$; $g_3'(x) = 1.3333333333333333*(4*x - 1) **(-0.6666666666666667)$, and at x = 1, $g_3'(x) \approx 0.456$ (< 1, best choice for convergence).

Based on these calculations, $g_3(x)$ is chosen for fixed-point iteration as it has the best convergence behavior.

c. Using **ONE** of the g(x) you identified as most likely to converge, perform 5 iterations starting from x_0 =1 and compute each step up to four decimal places. Provide your calculations in the form of a table to clearly show the convergence.

Ans.

Using $g_3(x)$, we perform 5 iterations starting from x0 = 1. Each result is rounded to 4 decimal places.

Iteration	X_n	$g(x_n)$
1	1	1.4422
2	1.7100	1.8008
3	1.9866	1.9080
4	2.0759	1.9402
5	2.1032	1.9498

d. What is the approximate root of this function?

After 5 iterations, the approximate root of $f(x) = x^3 - 4x + 1$ is approximately 2.1032.

[CLO 6: Approximation of root]

Q2: **[5 Marks]**

Assume you are given an implementation of newton ( ) method for finding the root of a function with the prototype as given below:

```
def newton(f, df, x0, tol = 1.e-6, maxit = 100):
    # f = the function f(x)
    # df = the derivative of f(x)
    # x0 = the initial guess of the solution
    # tol = tolerance for the absolute error
    # maxit = maximum number of iterations
```

Write the required python code to use this method to approximate the root of $f(x)=x^3-4x+1$, using an initial guess of 1.0, tolerance level of 0.0001 and a maximum of 10 iterations. Only provide the necessary code which you need to write to approximate the root of f(x). Do not modify the prototype of newton () method.

Ans.

```
def f(x):
    return x**3 - 4*x + 1

def df(x):
    return 3*x**2 - 4

x0 = 1.0
tol = 0.0001
maxit = 10

root = newton(f, df, x0, tol, maxit)
print(root)
```

[CLO 2: Interpolating a function via p(x)]

Q3: Hermite interpolating polynomials can be computed as follows: **[15 marks]**

> ☞ Given $N+1$ nodes $x_0 < x_1 < \cdots < x_N$ and the values $f(x_i)$ and $f'(x_i)$ for $i = 0, 1, \ldots, N$, the Hermite interpolating polynomial is the polynomial
>
> $$H_{2N+1}(x) = \sum_{i=0}^{N} [\alpha_i(x) f(x_i) + \beta_i(x) f'(x_i)] ,$$
>
> where α_i and β_i are given in terms of the Lagrange polynomials as
>
> $$\alpha_i(x) = [1 - 2\ell_i'(x_i)(x - x_i)]\ell_i^2(x) \quad \text{and} \quad \beta_i(x) = (x - x_i)\ell_i^2(x) .$$

Assume that a Python function `lagrange_basis(z,x)` has been written that computes Langrange polynomials given the interpolating data `x` and values z at which the polynomial is to be computed.

1. Employ `lagrange_basis` and `numpy` functions `polyder` and `poly1d` to write a Python function `Hermite(x,y,z)` that takes as inputs the interpolating data `x,y` and values `z` of the derivatives, and returns a Hermite interpolating polynomial H.

Ans.

```
def Hermite(x,y,z):
    N = len(x)
    p = np.poly1d([0])
    for i in range(N):
        L = lagrange_basis(x[i],x)
        dL = np.polyder(L)
        alpha = (np.poly1d([1])-2*dL(x[i])*np.poly1d([1,-x[i]]))*L**2
        beta = np.poly1d([1,-x[i]])*L**2
        p = p + alpha*y[i]+beta*z[i]
    return p
```

2. Write a piece of Python code that calls the function `Hermite` to compute the interpolating polynomial at `np.linspace(1, 5, 100)` for the interpolating data

xi	1	2	4	5
f(xi)	1	4	16	25
f'(xi)	2	4	8	10

Ans.

```
xi = np.array([1,2,4,5])
yi = np.array([1,4,16,25])
zi = np.array([2,4,8,10])
x = np.linspace(1,5,100)
p = Hermite(xi,yi,zi)
y = p(x)
```

[CLO 2: Interpolating a function via p(x)]

Q4: Consider the experimental data tabulated: **[15 marks]**

1. Fit a quadratic function to the data using `numpy` functions `polyfit` and `poly1d`.
2. Fit an exponential function to the data using `numpy` functions `polyfit` and `poly1d`.

t	y
0.09	15.1
0.32	57.3
0.69	103.3
1.51	174.6
2.29	191.5
3.06	193.2
3.39	178.7
3.63	172.3
3.77	167.5

Ans. 1

```
t = np.array([0.09,0.32,0.69,1.51,2.29,3.06,3.39,3.63,3.77])
y = np.array([15.1,57.3,103.3,174.6,191.5,193.2,178.7,172.3, 167.5])
aa = np.polyfit(t, y, 2)
yy = np.poly1d(aa)
```

Ans. 2

2.2:

```
z = np.log(y)
aa = np.polyfit(t, z, 1)
zz = np.poly1d(aa)
yy = np.exp(zz)
```

[CLO 3: Numerical Integrations]

Q5: Consider the $f(x) = e^x.\cos(x)$ in interval [-2,2] **[15 marks]**

a. Approximate the integral via composite midpoint quadrature rule for N = 8

b. Approximate the integral via gauss quadrature rule quadrature rule for N = 2, using below table

Gaussian nodes and weights for $N = 0, 1, 2, 3, 4$

N	z_i	w_i
0	0	2
1	$\pm\sqrt{\frac{1}{3}}$	1
2	$\pm\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
	0	$\frac{8}{9}$
3	$\pm\sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36}$

Ans: a.

$$h = \frac{b-a}{N} = \frac{2+2}{8} = 0.5$$

The points for 8 sub-intervals based on h values are given below.

[-2, -1.5, -1, -0.5,0, 0.5,1,1.5,2]

While the sub-intervals are

[-2, -1.5],[-1.5, -1],[-1, -0.5],[-0.5,0],[0, 0.5],[0.5,1],[1,1.5],[1.5,2]

Having the midpoints $-1.75, -1.25, -0.75, -0.25, 0.25, 0.75, 1.25, 1.75$. of each interval.

By applying the composite midpoint formula. The approximate solution is

$$\int_{-2}^{2} e^x \cos(x)\, dx \approx 0.5\,(f(-1.75) + f(-1.25) + f(-0.75) + f(-0.25) + f(0.25) + f(0.75) + f(1.25) + f(1.75))$$

$$\approx 0.5 \cdot \big(-0.03097 + 0.09034 + 0.34563 + 0.75459 + 1.24411 + 1.54899 + 1.10058 - 1.02574\big).$$

$$\approx 2.014$$

Ans b.

The Gaussian quadrature rule approximates the integral for N=2 by using the weights and roots from above table we can choose the weights and root accordingly,

$Z_0 = -\sqrt{\frac{3}{5}}$ $Z_1 = 0$ $Z_2 = \sqrt{\frac{3}{5}}$, $w_0 = \frac{5}{9}$ $w_1 = \frac{8}{9}$ $w_2 = \frac{5}{9}$

$$\int_{-2}^{2} e^x \cos(x)dx \approx w_0 f(x_0) + w_1 f(x_1) + w_2 f(x_2)$$

where z_i and w_i are the Gaussian nodes and weights for $N = 2$.

From the table provided:

Since this rule work in [-1,1] so by changing the variable

$$x_i = \frac{b-a}{2} \cdot z_i + \frac{a+b}{2}$$

$$X_0 = \frac{(-2-2)}{2}\left(-\sqrt{\frac{3}{5}}\right) + \frac{(-2-2)}{2} = -0.450$$

$$X_1 = \frac{(-2-2)}{2}(0) + \frac{(-2-2)}{2} = -2$$

$$X_2 = \frac{(-2-2)}{2}\left(\sqrt{\frac{3}{5}}\right) + \frac{(-2-2)}{2} = -3.549$$

Now apply the above to get final approximations.

$$\int_{-2}^{2} e^x \cos(x)dx \approx w_0 f(x_0) + w_1 f(x_1) + w_2 f(x_2)$$

By substituting the weights.

$$\frac{5}{9} f(-0.450) + \frac{8}{9} f(-2) + \frac{5}{9} f(-3.549)$$

$$\approx 0.318 + (-0.050) + (-0.0146)$$
$$\approx 0.2534.$$

c. Complete the following implementation for quadrature rule in **b, with N=2.**
Ans.

```
def g_quad(f, a, b):

    # define quadrature weights and nodes
    w = np.array([1,1])
    z = np.array([-np.sqrt(1/3), np.sqrt(1/3)])
    # implement formula
    c1 = (b-a)/2.0
    c2 = (a+b)/2.0
    s = c1*np.inner(w, f( c1*z + c2 ))
    return s
```

[CLO 3: Numerical Differentiations]

Q6: Consider the function $f(x)= x^5 +4x$, **[10 marks]**

a. Find the actual derivative at x= 3

Answer a.

Actual derivative is $5x^4+ 4$

At x=3

409.

b. Approximate the derivative at x=3 using a step size h=0.06, with the methods mentioned in the table on next page.

Using the following three formulae for approximations

Forward Difference:

$$f'(x) \approx \frac{f(x+h)-f(x)}{h}$$

Backward Difference:

$$f'(x) \approx \frac{f(x)-f(x-h)}{h}$$

Central Difference:

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h}$$

	Approximate value	Relative Error
Forward	425.53	4.04
Backward	393.12	3.88
Central	409.32	0.08

c. Complete the following implementation.

Ans.

```
def approximate_derivative(f, x, h, method):
    if method == "forward":
        return (f(x + h) - f(x)) / h
    elif method == "backward":
        return (f(x) - f(x - h)) / h
    elif method == "central":
        return (f(x + h) - f(x - h)) / (2 * h)
    else:
        raise ValueError("Method must be one of ['forward', 'backward', 'central']")

# Example function
def f(x):
    return x**5 + 4*x

# Test cases
print(approximate_derivative(f, 3, 0.06, method="forward"))  # Forward difference
print(approximate_derivative(f, 3, 0.06, method="backward"))  # Backward difference
print(approximate_derivative(f, 3, 0.06, method="central"))   # Central difference
```