AMSC 460 Homework Set 2 Due: 21.04.01

Note:

- Due: 2pm, April 01, 2021. Late submission is subject to automatic 20% reduction.
- When submitting, make sure you upload your matlab or python codes, in addition to your answer sheets to analytical questions.

Topic: Newton Interpolation with Divided Difference

Consider a polynomial interpolation problem $p_n(x)$ of an (unknown) function $f(x)$,

$$p_n(x) = \sum_{j=0}^{n} c_j \varphi_j(x) \qquad (1)$$

where c_j and $\varphi_j(x)$ are the j-th interpolating constant and function, respectively. Using (n+1) data sample $\{(x_j, y_j=f(x_j))\}$ for j=0,…,n, Newton interpolation using divided difference can be written as

$$c_j = f[x_0, \dots, x_j]$$
$$\varphi_j(x) = \prod_{i=0}^{j-1} (x - x_i) \qquad (2)$$

where

$$f[x_i] = y_i$$
$$f[x_0, \dots, x_j] = \frac{f[x_1, \dots, x_j] - f[x_0, \dots, x_{j-1}]}{x_j - x_0}$$

Main Problems

1. Show that the formula (2) is consistent with the data sample $\{(x_j, y_j)\}$

2. Write a pseudocode, as computationally efficient as possible (i.e., minimize number of operations)

3. Write a matlab or python code based on the pseudocode

4. For a data sample {(1,3),(5,11),(2,2),(4,12)},
   a. Obtain analytical solution, i.e., obtain c_j and $\varphi_j(x)$ by hand
   b. Show that the analytical solution is consistent with the data sample
   c. Predict $f(x^*)$ at $x^*=3$, i.e., compute it by hand

5. Using the same data sample
   a. Obtain c_j and $\varphi_j(x)$ by your matlab or python code, verify against your analytical solution obtained in Question 4 above
   b. Verify that the code reproduces the sample data
   c. Verify that the code reproduces $f(x^*=3)$ obtained in Question 4 above

Bonus points

B. Derive the formula (2) analytically.

$$c_j = f[x_0, \ldots, x_j]$$

$$\phi_j(x) = \prod_{i=0}^{j-1} (x - x_i)$$

where $f[x_i] = y_i$

$$f[x_0, \ldots, x_j] = \frac{f[x_1, \ldots, x_j] - f[x_0, \ldots, x_{j-1}]}{x_j - x_0}$$

given $\{(x_j, f(x_j))\}$ is

$$\begin{pmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_j(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_j(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \vdots & \ddots & & & \vdots \\ \phi_0(x_j) & \phi_1(x_j) & & & & \vdots \\ \vdots & \vdots & & & \ddots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \cdots & & & \phi_n(x_n) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

consistent?

plugging in $c_j = f[x_0, \dots, x_j]$

and $\phi_j(x) = \prod_{i=0}^{j-1} (x - x_i)$

$j=0$ $\quad \phi_0(x) = \prod_{i=0}^{-1} (x - x_i) = 1$

$$\phi_1(x) = x - x_0$$

$$\phi_j(x) = (x - x_0)(x - x_1) \cdots (x - x_{j-1})$$

$$\phi_n(x) = (x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & & & & & \vdots \\ \vdots & \vdots & & \ddots & & & \vdots \\ 1 & x_j - x_0 & \cdots & \cdots & (x-x_0)(x-x_1)\cdots(x-x_{j-1}) & & 0 \\ \vdots & \vdots & & & & \ddots & 0 \\ 1 & x_n - x_0 & \cdots & & (x-x_0)(x-x_1)\cdots(x-x_{n-1}) & & (x_n-x_0)(x_n-x_1)\cdots(x_n-x_{n-1}) \end{pmatrix} \begin{pmatrix} f[x_0] \\ f[x_0, x_1] \\ \vdots \\ f[x_0, x_1, \dots, x_j] \\ \vdots \\ f[x_0, \dots, x_j, \dots, x_n] \end{pmatrix} = \begin{pmatrix} f[x_0] \\ f[x_1] \\ \vdots \\ f[x_j] \\ \vdots \\ f[x_n] \end{pmatrix}$$

gives system of lin eqn.

$$f[x_0] = f[x_0] = y_0$$

$$f[x_0] + f[x_0, x_1](x_1 - x_0) = f[x_1]$$

$$\vdots$$

$$f[x_0] + \cdots f[x_0, \ldots, x_j](x - x_0) \cdots (x - x_{j-1})$$

$$= f[x_j]$$

$$= y_j$$

$$\vdots$$

$$f[x_0] + \cdots f[x_0, \ldots, x_j](x - x_0) \cdots (x - x_{j-1})$$

$$+ \cdots f[x_0, \ldots, x_n](x - x_0) \cdots (x - x_n)$$

$$= f[x_n] = y_n$$

which can be solved w/ sample data by

plugging eqn for $i=0$ into $i=1, \ldots, i=j-1$ into $i=j, \ldots$ $i=1, \ldots, i=j, \ldots i=n-1$ into $i=n$.

$\Rightarrow \forall j,$

$$p_0(x_0) = y_0$$

$$p_j(x_j) = p_0(x_0) + \ldots + c_j q_j(x_j)$$

$$= y_j$$

and so on.

2) given $\vec{x}$ a n+1 length list. (Input).
$\vec{y}$ a n+1 length list.

Let A be $(n+1) \times (n+1)$ matrix
(indexed at 0)

for $i = 0$ to n

$A_{i0} = y_i$

end

for $j = 1$ to n

for $i = 0$ to $n-j$

$A_{ij} = (A_{i+1,j-1} - A_{i,j-1}) / x_{i+j} - x_i$

end

end

return $A_{0,0:n}$ (first row) as A

Need to create basis functions.

let phi be a lenght n+1 list

for $i = 1$ to n

phi[i] = $-x_{i-1}$

end

phi{0} = 1

return phi

now the fun part of algo

given x_* (Input).

for i=1 to n

phi[i] = phi{i} + x_*

end

for i=1 to n

phi{i} = phi(i}'phi[i-1}

end

Let p = 0

for i=0 to n

p = p + phi[i]*A[i]

end

return p (which is $f(x^*)$).

i	x	y
0	1	3
1	5	11
2	2	2
3	4	12

4). $c_0 = 3$ $\boxed{\phi_0(x) = 1}$

$$c_1 = f[x_0, x_1]$$

$$= \frac{f[x_1] - f[x_0]}{x_1 - x_0}$$

$$= \frac{8}{4} = 2$$

$$\boxed{\phi_1(x) = x - 1}$$

$$c_2 = f[x_0, x_1, x_2]$$

$$= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

$$f[x_1, x_2] = \frac{f[x_2] - f[x_1]}{x_2 - x_1} = 3$$

$$c_2 = \frac{3 - 2}{1} = 1$$

$$\boxed{\phi_2(x) = (x-1)(x-5)}$$

$$c_3 = f[x_0, x_1, x_2, x_3]$$

$$= \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0}$$

$$f[x_1, x_2, x_3] = \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1}$$

$$f[x_2, x_3] = \frac{f[x_3] - f[x_2]}{x_3 - x_2} = 5$$

$$f[x_1, x_2, x_3] = \frac{5 - 3}{-1} = -2$$

$$c_3 = \frac{-2 - 1}{4 - 1} = -1$$

$$\boxed{\phi_3(x) = (x-1)(x-5)(x-2)}$$

$$\sigma(x) = p_3(x) = c_0\phi_0 + c_1\phi_1 + c_2\phi_2 + c_3\phi_3$$

phi's in green boxes.

$c_0 = 3, \; c_1 = 2, \; c_2 = 1, \; c_3 = -1$

$$\Phi_0(x^*) = 1$$

$$\Phi_1(x^*) = 2$$

$$\Phi_2(x^*) = 2(-2) = -4$$

$$\Phi_3(x^*) = -4(1) = -4$$

$$P_3(x^*) = 3 + 4 + (-4) + 4$$

$$= 7$$

$$\sigma(1) = 3 + 0 + 0 + 0$$

$$\sigma(5) = 3 + 2(5-1) + 0 + 0 = 11$$

$$\sigma(2) = 3 + 2(2-1) + (2-1)(2-5) + 0$$
$$= 3 + 2 + (-3) = 2$$

$$\sigma(4) = 3 + 2(4-1) + (4-1)(4-5)$$
$$- (4-1)(4-5)(4-2)$$

$$= 3 + 6 - 3 + 6 = 12$$

The above work is verified w/ the same integers produced in the code.

HW2

April 1, 2021

[1]:
```python
import numpy as np

def get_coeff(x: list, y: list) -> np.ndarray:
    #This will do divided differences
    A = np.zeros((len(x),len(y)), dtype=float)
    for row_loc in range(A.shape[0]):
        A[row_loc,0] = y[row_loc]
    for col_loc in range(1, A.shape[0]):
        for row_loc in range(A.shape[0] - col_loc):
            A[row_loc,col_loc] = (A[row_loc+1,col_loc-1] -␣
 ↪A[row_loc,col_loc-1])/(x[row_loc+col_loc] - x[row_loc])
    #coeff are stored along the first row of the array
    return A[0,:]

def make_phi(x: list) -> np.ndarray:
    phi = np.zeros((len(x),))
    for idx in range(1, len(x)):
        phi[idx] = -x[idx-1]
    #first basis function for a polynomial is always degree 0
    phi[0] = 1
    return phi

def interpolate_x_star(coeff: np.ndarray, phi: np.ndarray, x_star):
    phi[1:] = phi[1:] + x_star
    for basis in range(1, phi.shape[0]):
        phi[basis] = phi[basis]*phi[basis-1]
    sol = coeff.dot(phi)
    print("phi is: " + str(phi))
    print("coeff are: " + str(coeff))
    return sol
```

[2]:
```python
def solve_newton_interpolation(x: list, y: list, x_star: int) -> float:
    return interpolate_x_star(get_coeff(x,y), make_phi(x), x_star)
```

[3]:
```python
x = [1, 5, 2, 4]
y = [3, 11, 2, 12]
x_star = 3
```

```
ans = solve_newton_interpolation(x, y, 3)
print("f("+str(x_star)+")=" + str(ans) + "\n")
```

```
phi is: [ 1.  2. -4. -4.]
coeff are: [ 3.  2.  1. -1.]
f(3)=7.0
```

[4]:

```
for x_star in x:
    print("f("+str(x_star)+")=" + str(solve_newton_interpolation(x, y,␣
 ↪x_star))+"\n")
```

```
phi is: [ 1.  0. -0.  0.]
coeff are: [ 3.  2.  1. -1.]
f(1)=3.0

phi is: [1. 4. 0. 0.]
coeff are: [ 3.  2.  1. -1.]
f(5)=11.0

phi is: [ 1.  1. -3. -0.]
coeff are: [ 3.  2.  1. -1.]
f(2)=2.0

phi is: [ 1.  3. -3. -6.]
coeff are: [ 3.  2.  1. -1.]
f(4)=12.0
```

[ ]:

```

```