



React Web Application

จัดทำโดย

64-040626-3035-0 นายเดชา บุญมาพาทรัพย์

คณะวิทยาศาสตร์ประยุกต์ สาขาวิทยาการคอมพิวเตอร์

มหาวิทยาลัยพระจอมเกล้าเจ้าอยู่หัว

รายงานนี้เป็นส่วนหนึ่งของวิชา การโปรแกรมเชิงวัตถุ (Object-Oriented Programming)

รหัสวิชา 040613204 ภาคเรียนที่ 1 ปีการศึกษา 2565

คำนำ

รายงานนี้เป็นส่วนหนึ่งของรายวิชา Object-Oriented Programming รหัสวิชา 040613204 ภาคเรียนที่ 1 ปีการศึกษา 2565 โดยอธิบายถึงหลักการทำงานของ React Web Application ที่ได้ทำในวิชา Numerical Method รหัสวิชา 040613105 ที่จัดทำเพื่อการเรียนรู้และศึกษาวิธีการเขียนโค้ดด้วยภาษา JavaScript และ HTML รวมไปถึงการใช้ Library จากภายนอกเช่น math.js, json-server เป็นต้น

ทั้งนี้ ทางผู้จัดทำคาดหวังว่ารายงานฉบับนี้จะเป็นประโยชน์สำหรับผู้ผู้อ่านหรือสนใจสามารถเข้าใจหลักการทำงานของ Web Application นี้ หากมีข้อผิดพลาดประการใด ผู้จัดทำขออภัย ณ ที่นี้

เดชา บุญมาพาทรัพย์

ผู้จัดทำ

สารบัญ

เรื่อง

หน้า

ROOT OF EQUATION

- หน้าแรกของ Web 1-7
- การคำนวณด้วยวิธี Bisection 8-15
- การดึงค่าจาก REST API ด้วย json-server 16-21

LINEAR ALGEBRAIC EQUATION

- การคำนวณด้วยวิธี Matrix Inversion 22-27

INTERPOLATION

- การคำนวณด้วยวิธี Lagrange Interpolation 28-31

REGRESSION

- การคำนวณด้วยวิธี Linear และ Polynomial Regression 32-39

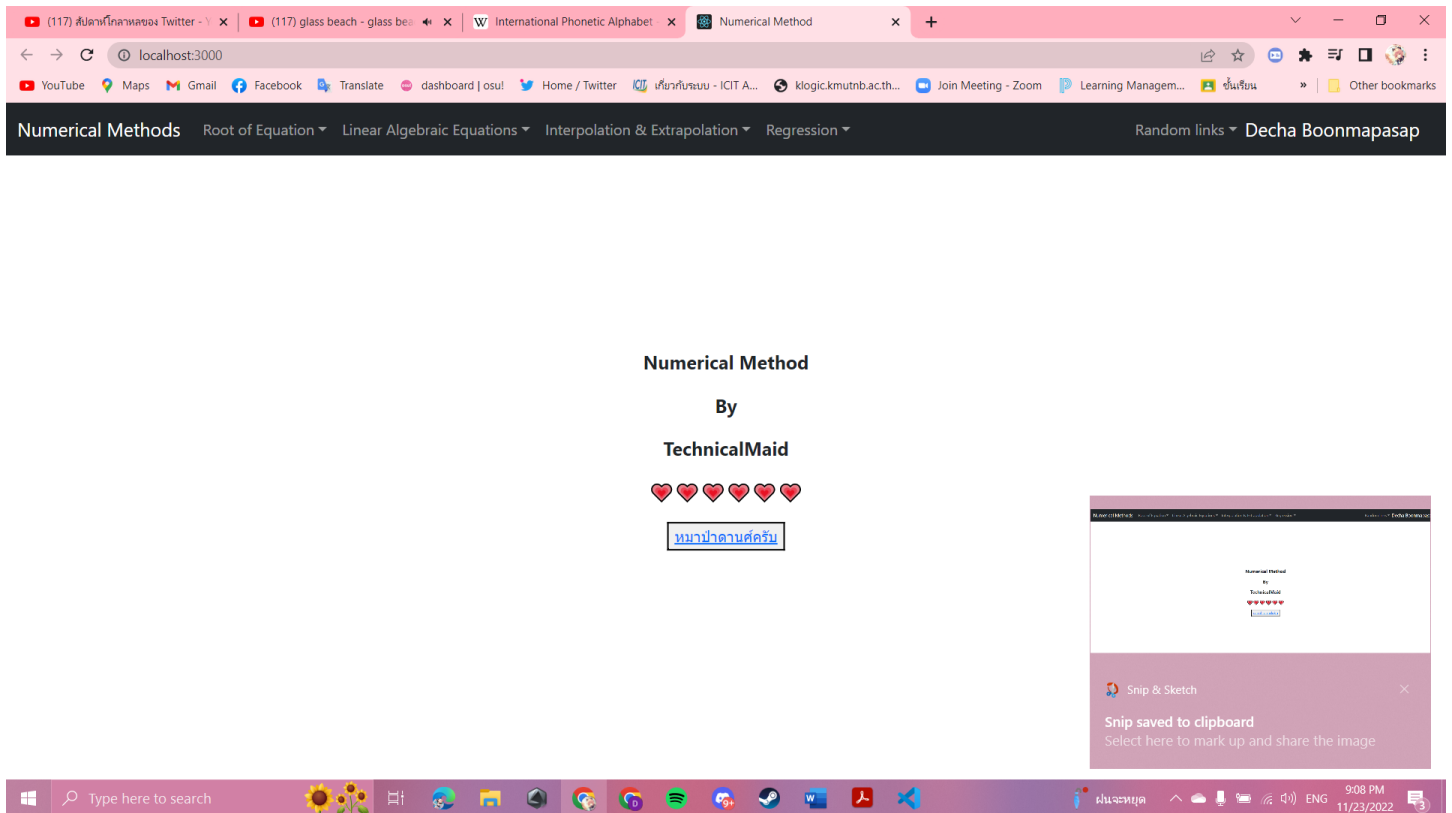
ROOT OF EQUATION

1. หน้าแรกของ Web

ทำการรัน Client ขึ้นมาโดยใช้คำสั่ง “npm start”

```
PS C:\Users\ASUS\Desktop\Documents\Numerical_Project\my-app> npm start  
  
> my-app@0.1.0 start  
> react-scripts start
```

เมื่อทำการรันแล้วจะทำการเรียกหน้าเว็บที่เป็น Default



หลักการทำงานในหน้านี้

```
JS index.js  X
src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import './index.css';
4  import App from './App';
5  import reportWebVitals from './reportWebVitals';
6
7
8  const root = ReactDOM.createRoot(document.getElementById('root'));
9  root.render(
10   <React.StrictMode>
11     <App />
12   </React.StrictMode>
13 );
14
15 // If you want to start measuring performance in your app, pass a function
16 // to log results (for example: reportWebVitals(console.log))
17 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
18 reportWebVitals();
19
```

โดยปกติแล้ว react จะไปเรียกหน้า index.js เป็นหน้าแรกซึ่งหน้านี้จะบังคับดึง element id 'root'

แล้วทำการ Render Component ชื่อว่า App (App.js)

โดย Import แต่ละ Component มาที่หน้านี้

```

JS index.js X JS App.js 2, M X
src > JS App.js > ...
1 import './App.css';
2 import './css/styles.css';
3 import 'bootstrap';
4 import React from "react";
5
6 import { BrowserRouter, Route, Routes, Switch } from "react-router-dom";
7 //import Bisection from './js/Bisection'
8 //import Falseposition from './js/False-position'
9 //import OnePoint from './js/One-point-Iteration'
10 //import Newton from './js/Newton-Raphson'
11 //import Secent from './js/Secent'
12
13 |
14 import Layout from "./Layout"
15 import Home from "./Home"
16
17 import Bisection from './Root-of-Equation/Bisection';
18 import Falseposition from './Root-of-Equation/False-position';
19 import Newton from './Root-of-Equation/Newton';
20 import Onepoint from './Root-of-Equation/Onepoint';
21 import Secent from './Root-of-Equation/Secent';
22
23 import Cramer from './Linear-Algebraic-Equations/Cramer-rule'
24 import GuessElim from './Linear-Algebraic-Equations/Guess-Elim';
25 import GuessJordan from './Linear-Algebraic-Equations/Guess-Jordan';
26 import MatrixInversion from './Linear-Algebraic-Equations/Matrix-Inversion';
27 import LUdecomposition from './Linear-Algebraic-Equations/LU-Decomposition';
28 import CholeskyDecomposition from './Linear-Algebraic-Equations/Cholesky-Decomposition';
29 //
30 import Jacobi from './Linear-Algebraic-Equations/Jacobi';
31 import GuessSeidel from './Linear-Algebraic-Equations/GaussSeidel';
32
33 import NewtonDevided from './Interpolation-and-Extrapolation/Newton-devided';
34 import Lagrange from './Interpolation-and-Extrapolation/Lagrange';
35 import Sprine from './Interpolation-and-Extrapolation/Sprine';
36
37 import Regression from './Regression/Regression';
38
39 v function App() {
40     const handleSelect = (eventKey) => alert(`${eventKey}`); //คลิกไปจ้ง
41     return (
42         <BrowserRouter>
43         <Routes>
44
45             <Route path="/" element={<Layout />} />
46             <Route index element={<Home />} />
47             <Route path="Bisection" element={<Bisection />} />
48             <Route path="Falseposition" element={<Falseposition />} />
49             <Route path="Onepoint" element={<Onepoint />} />
50             <Route path="Newton" element={<Newton />} />
51             <Route path="Secent" element={<Secent />} />
52
53
54             <Route path="Cramer" element={<Cramer />} />
55             <Route path="GuessElim" element={<GuessElim />} />
56             <Route path="GuessJordan" element={<GuessJordan />} />
57             <Route path="MatrixInversion" element={<MatrixInversion />} />
58             <Route path="LUdecomposition" element={<LUdecomposition />} />
59             <Route path="CholeskyDecomposition" element={<CholeskyDecomposition />} />
60
61             <Route path="Jacobi" element={<Jacobi />} />
62             <Route path="GuessSeidel" element={<GuessSeidel />} />
63
64
65             <Route path="NewtonDevided" element={<NewtonDevided />} />
66             <Route path="Lagrange" element={<Lagrange />} />
67             <Route path="Sprine" element={<Sprine />} />
68
69             <Route path="Regression" element={<Regression />} />
70         </Routes>
71     </BrowserRouter>
72
73
74
75
76 );
77 }
78
79 export default App;

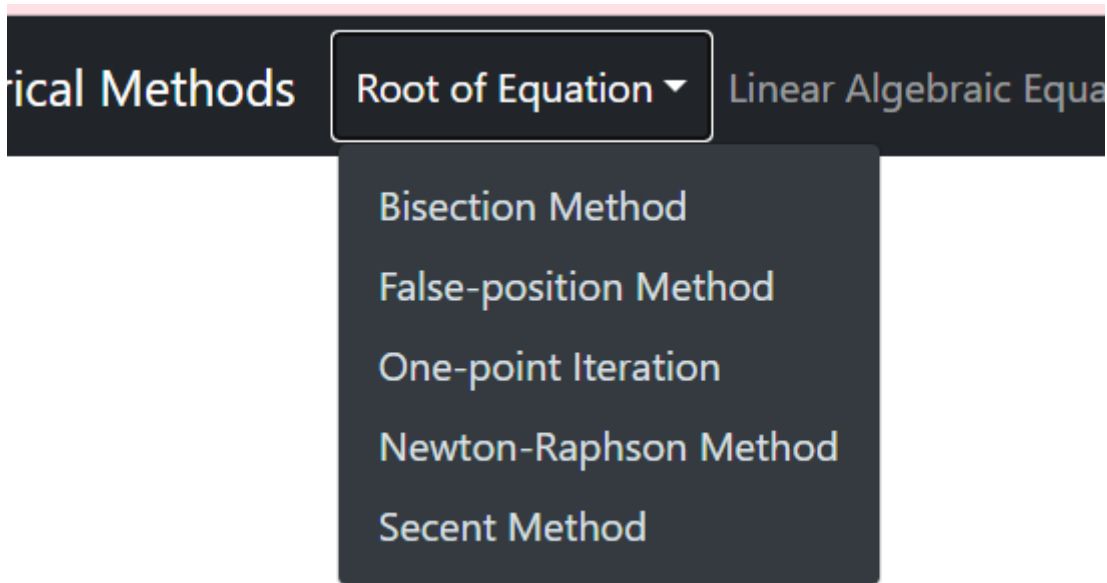
```

ใน Component App ก็คือมีการเก็บ Path ของ Component แต่ละตัว โดยทุกๆหน้าจะให้ทำการเรียก Component Layout ออกมา

```
45 <Route path="/" element={<Layout />}>
```

ใน Component Layout (Layout.js) จะมีการนำ bootstrap มาใช้เพื่อเขียนเป็น Navbar โดยกำหนด path ให้แต่ละปุ่มว่า กดคลิกแล้วให้ไปเรียก Component อะไร

```
JS Layout.js 2 X
src > JS Layout.js > [1] Layout
1 import { Outlet, Link } from "react-router-dom";
2 import Container from 'react-bootstrap/Container';
3 import Nav from 'react-bootstrap/Nav';
4 import Navbar from 'react-bootstrap/Navbar';
5 import NavDropdown from 'react-bootstrap/NavDropdown';
6 import Button from 'react-bootstrap/Button';
7 import './css/styles.css';
8
9 const Layout = () => {
10   return (
11     <>
12       <Navbar collapseOnSelect variant="dark" bg="dark" expand="lg">
13         <Container fluid>
14           <Navbar.Brand href="/">Numerical Methods</Navbar.Brand>
15           <Navbar.Toggle aria-controls="navbar-dark-example" />
16           <Navbar.Collapse id="navbar-dark-example">
17             <Nav>
18               <NavDropdown
19                 id="nav-dropdown-dark-example"
20                 title="Root of Equation"
21                 menuVariant="dark"
22               >
23                 <NavDropdown.Item href="/Bisection">Bisection Method</NavDropdown.Item>
24                 <NavDropdown.Item href="/Falseposition">False-position Method</NavDropdown.Item>
25                 <NavDropdown.Item href="/Onepoint">One-point Iteration</NavDropdown.Item>
26                 <NavDropdown.Item href="/Newton">Newton-Raphson Method</NavDropdown.Item>
27                 <NavDropdown.Item href="/Secant">Secant Method</NavDropdown.Item>
28               </NavDropdown>
29             </Nav>
30           <Nav>
```



ยกตัวอย่างเช่นหาก กดที่ Root of Equation ก็จะแสดงเป็น Dropdown มาโดยแต่ละตัวเลือกก็ทำการชี้ path เช่น หากกด Bisection Method ก็ทำการเปลี่ยน Path ไปที่ /Bisection แล้วถ้า path เป็น /Bisection ก็จะทำการไปเรียก Component ที่ชื่อ Bisection

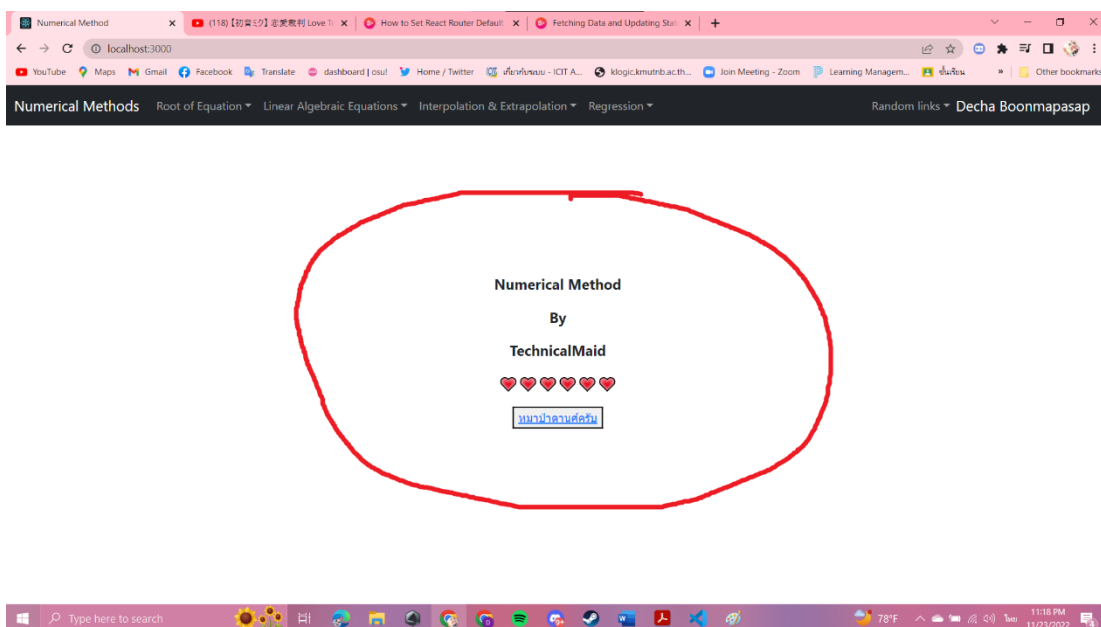
ลำดับต่อไป หน้า App.js จะทำการเรียก Component Home (Home.js)

```
46 | <Route index element={<Home />} />|
```

ภายใน Home.js

```
JS Home.js X
src > JS Home.js > [0] default
1  const Home = () => {
2    return <div><div class="container">
3      <div class="text-center mt-5">
4        <div class="text-center fw-bold text-dark position-absolute top-50 start-50 translate-middle fs-5">
5          <p></p>
6          <p>Numerical Method</p>
7          <p>By</p>
8          <p>TechnicalMaid</p>
9          <p>&#128151;&#128151;&#128151;&#128151;&#128151;&#128151;&#128151;</p>
10         <p></p>
11         <button type="button" class="top-50 start-50 fs-6" align="center">
12           <a href="https://www.youtube.com/watch?v=uMPA1bXhBxE">หน้าบ้านสัปดาห์</a>
13         </button>
14       </div>
15     </div>
16   </div>
17 </div>
18 };
19
20 export default Home;
```

Component Home จะแสดงหน้าตาแรกขึ้นมา

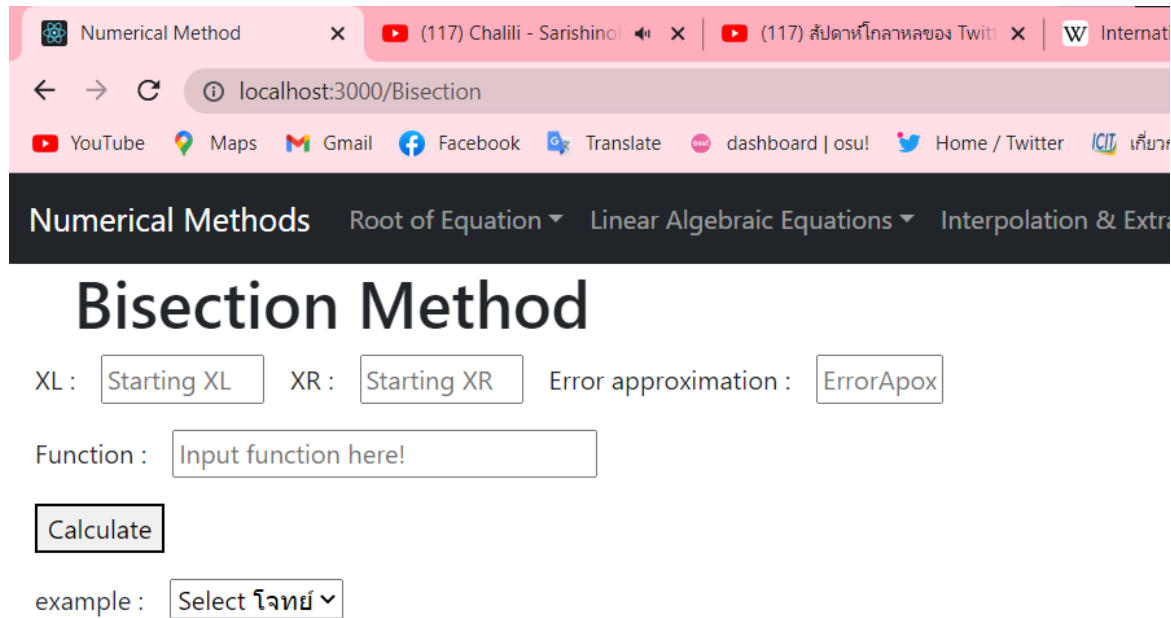


และส่วนสุดท้ายในหน้าแรกคือ

```
47 <Route path="Bisection" element={<Bisection />} />
48 <Route path="Falseposition" element={<Falseposition />} />
49 <Route path="Onepoint" element={<Onepoint />} />
50 <Route path="Newton" element={<Newton />} />
51 <Route path="Secent" element={<Secent />} />
52
53
54 <Route path="Cramer" element={<Cramer />} />
55 <Route path="GuessElim" element={<GuessElim />}/>
56 <Route path="GuessJordan" element={<GuessJordan />}/>
57 <Route path="MatrixInversion" element={<MatrixInversion />}/>
58 <Route path="LUDecomposition" element={<LUDecomposition />}/>
59 <Route path="CholeskyDecomposition" element={<CholeskyDecomposition />}/>
60
```

สร้าง path เชื่อมกับ ทุก Component ที่เขียนมา

2. การคำนวณด้วยวิธี Bisection



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/Bisection'. The page title is 'Numerical Method'. Below the title, there is a navigation bar with links: 'Numerical Methods', 'Root of Equation', 'Linear Algebraic Equations', and 'Interpolation & Extrapolation'. The main heading is 'Bisection Method'. Below the heading, there are input fields for 'XL : Starting XL', 'XR : Starting XR', 'Error approximation : ErrorApox', and 'Function : Input function here!'. There is a 'Calculate' button and an 'example : Select โจทย์' dropdown menu.

หลังจากที่เรากดปุ่ม Root of Equation -> Bisection Method จะเป็นการเรียก Component ที่มีชื่อว่า Bisection (Bisection.js)

แล้วเปลี่ยน Path เป็น /Bisection

ใน Component Bisection นี้จะมีการแสดงกล่อง Form สำหรับ Input มาก่อน

ถ้ามีค่า XL, XR, ErrorApox และ Funct ถูกใส่และมีการกดปุ่ม Calculate จะมีการส่งค่าที่เก็บมาไปยังฟังก์ชันที่มีชื่อว่า getValue

```

169     return(<body>
170         <div>
171             <form onSubmit={getValue}>
172                 <div>
173                     <h1>&nbsp;Bisection Method&nbsp;</h1>
174                     <label htmlFor='XL'>&nbsp;XL :&nbsp;</label>
175                     <input
176                         id='XL'
177                         name='XL'
178                         placeholder='Starting XL'
179                         value={getXL}
180                         onChange={event => setXL(event.target.value)}
181                         size='8'
182                     />
183                     <label htmlFor='XR'>&nbsp;XR :&nbsp;</label>
184                     <input
185                         id='XR'
186                         name='XR'
187                         placeholder='Starting XR'
188                         value={getXR}
189                         onChange={event => setXR(event.target.value)}
190                         size='8'
191                     />
192                     <label htmlFor='ErrorApox'>&nbsp;Error approximation :&nbsp;</label>
193                     <input
194                         id='ErrorApox'
195                         name='ErrorApox'
196                         placeholder='ErrorApox'
197                         value={getErrorApox}
198                         onChange={event => setErrorApox(event.target.value)}
199                         size='5'
200                     />
201                 </div>
202                 <p></p>
203                 <div>
204                     <label htmlFor='Funct'>&nbsp;Function :&nbsp;</label>
205                     <input
206                         id='Funct'
207                         name='Funct'
208                         placeholder='Input function here!'
209                         value={getFunct}
210                         onChange={event => setFunct(event.target.value)}
211                         size='30'
212                     />
213                 </div>
214                 <p></p>
215                 <p>
216                     <div>
217                         &nbsp;<button>Calculate</button>
218                     </div>
219                 </p>
220             </form>

```

```

11  const Bisection = () => {
12      var Funct,ErrorApox,XL,XR;
13
14      var [getFunct, setFunct] = useState('')
15      var [getErrorApox, setErrorApox] = useState('')
16      var [getXL, setXL] = useState('')
17      var [getXR, setXR] = useState('')
18      var xmgraph = xmarray;
19      var igrph = iarray;
20

```

```

100  var getValue = e => { //hale input event and pass value to function
101      e.preventDefault();
102      Funct = getFunct
103      ErrorApox = getErrorApox;
104      XL = getXL;
105      XR = getXR;
106      console.log(XL);
107      console.log(XR);
108      console.log(ErrorApox);
109      console.log(Funct);
110      iarray.splice(0,iarray.length) //clear array everytime user click calculate
111      xmarray.splice(0,xmarray.length)
112
113      BisectionCalcFunction(XL,XR,ErrorApox,Funct)
114  }

```

ฟังก์ชัน `getValue` จะทำการเก็บตัวแปรที่กรอกข้อมูลมาเป็น ตัวแปร global variable

แล้วทำการเรียกฟังก์ชัน `BisectionCalcFunction` โดยส่งค่า `XL,XR,ErrorApox` และ `Funct` เข้าไป

อธิบายหลักการทำงานของ BisectionCalcFunction

ภายใน ฟังก์ชัน BisectionCalcFunction()

```
116 function BisectionCalcFunction(XL,XR>ErrorApox,Funct)
117 {
118     var i = 0;
119     var x1 = parseFloat(XL);
120     var xr = parseFloat(XR);
121     var xm,xold;
122     var ErrorApox_Answer=10000000; //set as default
123     var inputerrorapox = parseFloat(ErrorApox)
124     let text = "";
125     let finalanswer = "===>";
126
127     function fx(input)
128     {
129         const exprfx = math.parse(Funct) //parse to math expression
130         return exprfx.evaluate({x: input}); //eval the expression to input for example if function is x^2-7 -> input^2-7
131     }
132
133     if(xl!=null && xr!=null && Funct!=null && inputerrorapox!=null){//bisection function
134         while(ErrorApox_Answer>inputerrorapox && i!=100)
135         {
136             xm=(x1+xr)/2;
137             if(fx(xm)*fx(xr)<0)
138             {
139                 xold=x1
140                 x1=xm
141             }
142             if(fx(xm)*fx(xr)>0)
143             {
144                 xold=xr
145                 xr=xm
146             }
147             ErrorApox_Answer = Math.abs((xm-xold)/xm)*100
148             i++
149             xmarray.push(xm.toFixed(6));
150             iarray.push(i) //push to store in array (use for render graph)
151             console.log("XL = "+x1) //console log for debugging
152             console.log("XM = "+xm)
153             console.log("XR = "+xr)
154             console.log("Errorapox = "+ErrorApox_Answer)
155             text = text+"At Iteration #"+i+" XM = "+xm.toFixed(6)+" with Errorapox of "+ErrorApox_Answer.toFixed(6)+"<br>"
156         }
157         finalanswer = finalanswer+"XM value is "+xm.toFixed(6)+" at Iteration #"+i+"<br>";
158
159         document.getElementById("finalans").innerHTML = finalanswer
160         console.log(finalanswer)
161         console.log(xmarray)
162         console.log(iarray)
163         document.getElementById("finaltext").innerHTML = text
164         document.getElementById("finalxm").innerHTML = xmarray //pass elementID
165     }
166 }
167 }
```

- ในส่วนบรรทัดที่ 118-125 จะเป็นค่าเริ่มแรก
- วิธีการทำ **Bisection** คือจะนำค่าฝั่งซ้าย+ค่าฝั่งขวา แล้วหาร 2 เพื่อหาค่ากลาง (ตามโค้ดบรรทัดที่ 137) แล้วทำการเช็คค่า $F(\text{ค่ากลาง}) * F(\text{ค่าฝั่งขวา})$ ว่ามากกว่าหรือน้อยกว่า 0
- การหา $F(x)$ ของแต่ละตัวจะทำการเอาค่า x ไปแทนในสมการ (สมมุติถ้า Function เป็น x^2-7 ค่า $F(2)$ ก็คือ $2^2-7 = -3$ และ ค่า $F(3)$ คือ $3^2-7 = 2$)
ในส่วนนี้ต้องใช้ **Math.js** มาช่วยโดยการเปลี่ยน Function ที่เป็น String ให้แทน x เข้าไปแล้วเป็นคำตอบ (function ย่อยในบรรทัดที่ 127-131)
- โดยถ้ามากกว่า 0 ให้แทนค่ากลางเป็นค่าฝั่งขวา ถ้าน้อยกว่า 0 ให้แทนค่ากลางเป็นค่าฝั่งซ้าย (If statement ในบรรทัดที่ 138-147)
- จากนั้นหาค่า **Error** ของแต่ละรอบโดยการนำ
 $|(\text{ค่ากลางรอบปัจจุบัน}-\text{ค่ากลางรอบที่แล้ว}) / \text{ค่ากลางรอบปัจจุบัน}| * 100$ (บรรทัดที่ 148)
- ทำการเก็บค่ากลางที่ได้มาแต่ละรอบเก็บไว้ใน **Array** ตัวนี้ เพื่อใช้สำหรับการ **Plot** เป็นกราฟ (บรรทัดที่ 150)
- ทำการวนซ้ำไปเรื่อยๆจนกว่า I (รอบ iteration) จะเกิน 100 หรือ **Error** ที่ได้จากคำตอบเกินจาก **Input ErrorApox** ที่รับมา (เงื่อนไข While loop บรรทัดที่ 135)
- ปรี้นคำตอบสุดท้ายเหมือนหลุด while loop (บรรทัด 158)
- จากนั้นแสดงคำตอบทั้งหมดโดยการฝังโค้ด **HTML** แล้วทำการเพิ่ม **String** เข้าไปใน ID ต่างๆ (บรรทัดที่ 156-165)

```

● 232      <h2><p id = 'finalans'></p></h2>
233      <p id = 'chart'></p>
234      <p id = 'finaltext'></p>

```

Bisection Method

XL : XR : Error approximation :

Function :

example :

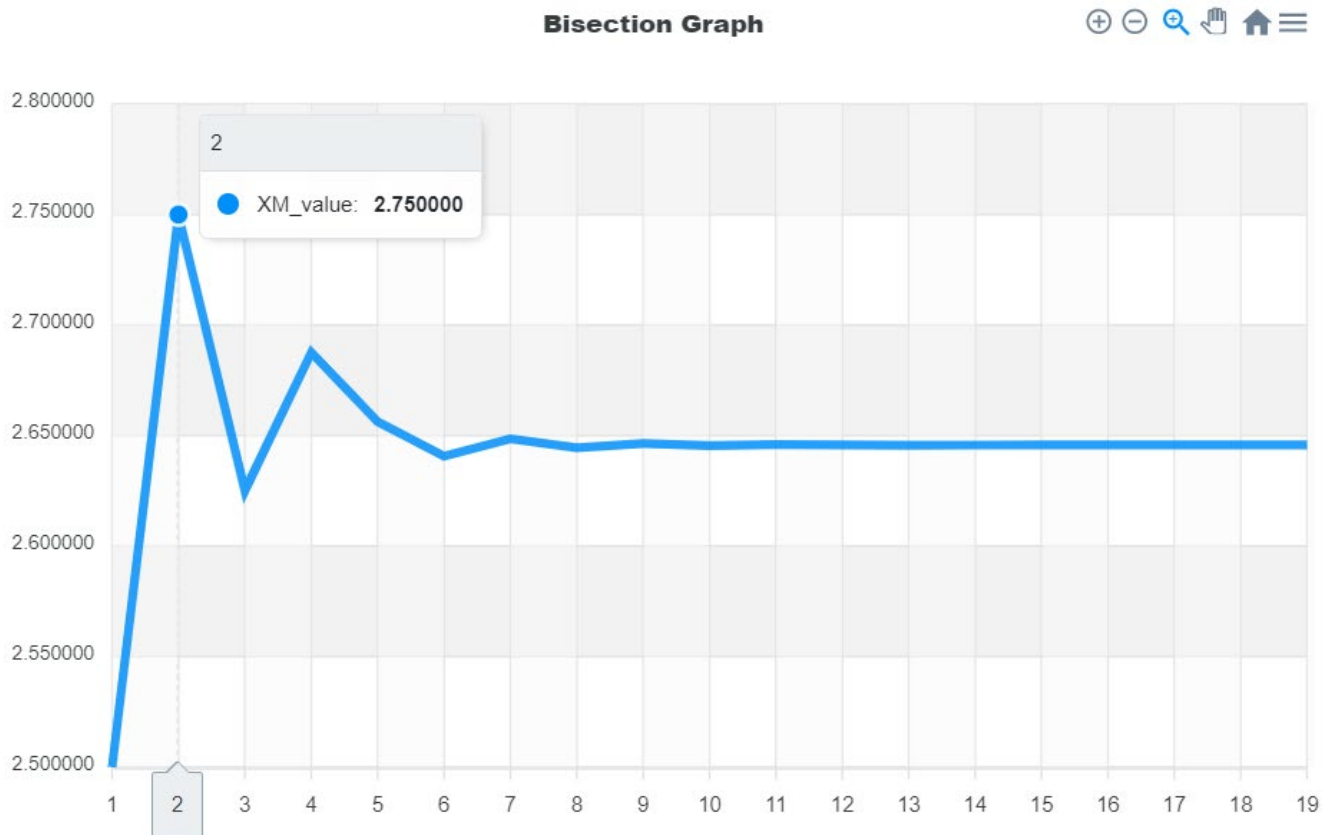
==>XM value is 2.645750 at Iteration #19

At Iteration #1 XM = 2.500000 with Errorapox of 20.000000
At Iteration #2 XM = 2.750000 with Errorapox of 9.090909
At Iteration #3 XM = 2.625000 with Errorapox of 4.761905
At Iteration #4 XM = 2.687500 with Errorapox of 2.325581
At Iteration #5 XM = 2.656250 with Errorapox of 1.176471
At Iteration #6 XM = 2.640625 with Errorapox of 0.591716
At Iteration #7 XM = 2.648438 with Errorapox of 0.294985
At Iteration #8 XM = 2.644531 with Errorapox of 0.147710
At Iteration #9 XM = 2.646484 with Errorapox of 0.073801
At Iteration #10 XM = 2.645508 with Errorapox of 0.036914
At Iteration #11 XM = 2.645996 with Errorapox of 0.018454
At Iteration #12 XM = 2.645752 with Errorapox of 0.009228
At Iteration #13 XM = 2.645630 with Errorapox of 0.004614
At Iteration #14 XM = 2.645691 with Errorapox of 0.002307
At Iteration #15 XM = 2.645721 with Errorapox of 0.001153
At Iteration #16 XM = 2.645737 with Errorapox of 0.000577
At Iteration #17 XM = 2.645744 with Errorapox of 0.000288
At Iteration #18 XM = 2.645748 with Errorapox of 0.000144
At Iteration #19 XM = 2.645750 with Errorapox of 0.000072

อธิบายหลักการทำงานการ Plot คำตอบที่ได้ออกมาเป็น Graph

```
23  var options = { //graph related
24      chart: {
25          type: 'line',
26          width: '750'
27      },
28      series: [{
29          name: "XM_value",
30          data: xmgraph
31      }],
32      xaxis: {
33          categories: igrph
34      },
35      grid: {
36          row: {
37              colors: ['#e5e5e5', 'transparent'],
38              opacity: 0.5
39          },
40          column: {
41              colors: ['#f8f8f8', 'transparent'],
42          },
43          xaxis: {
44              lines: {
45                  show: true
46              }
47          }
48      },
49      title: {
50          text: 'Bisection Graph',
51          align: 'center',
52          margin: 10,
53          offsetX: 0,
54          offsetY: 0,
55          floating: false
56      }
57  }
58
59  var chart = new ApexCharts(document.querySelector("#chart"), options);
60  chart.render(); //render chart (every time that state change)
61
```

- ทำการดึงค่ากลางที่ได้มาแต่ละรอบ มาใส่ในแกน y (บรรทัดที่ 28-30)
- ทำการดึงค่ารอบมาใส่ในแกน x (บรรทัดที่ 32-33)
- ทำการวาดกราฟโดยใช้ข้อมูลดังกล่าว (บรรทัดที่ 59-60)



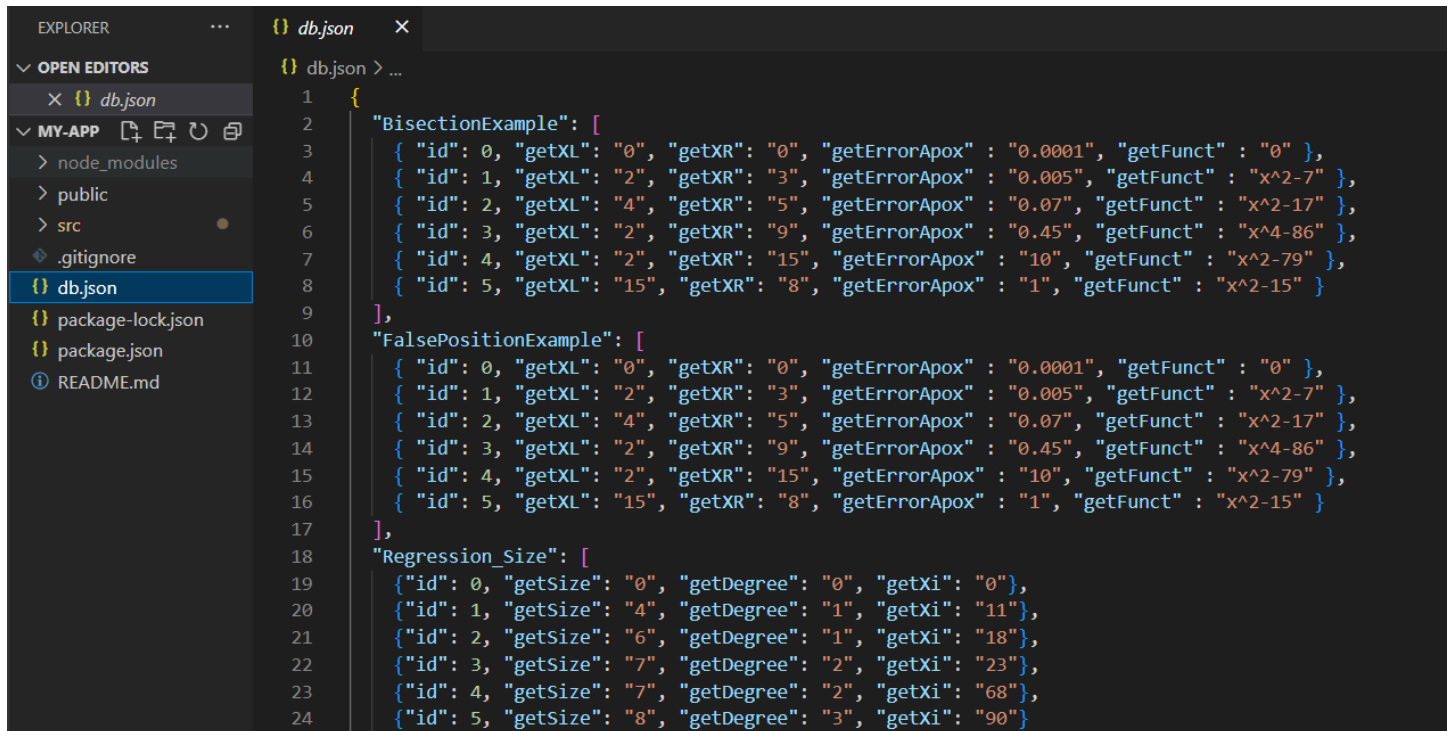
3. การดึงค่าจาก API ด้วย json-server

เริ่มแรกทำการลง library ชื่อ json-server ด้วยคำสั่ง “npm install -g json-server”

```
PS C:\Users\ASUS\Desktop\Documents\Numerical_Project\my-app> npm install -g json-server
```

จากนั้นทำการสร้างไฟล์มาไฟล์หนึ่งเป็น .json

ในที่นี้ผู้จัดทำสร้างไฟล์ที่ชื่อว่า db.json ขึ้นมา



The screenshot shows a VS Code editor with a file named `db.json` open. The file contains a JSON object with three arrays: `BisectionExample`, `FalsePositionExample`, and `Regression_Size`. Each array contains objects with various attributes like `id`, `getXL`, `getXR`, `getErrorApox`, `getFunct`, `getSize`, `getDegree`, and `getXi`.

```
{
  "BisectionExample": [
    { "id": 0, "getXL": "0", "getXR": "0", "getErrorApox": "0.0001", "getFunct": "0" },
    { "id": 1, "getXL": "2", "getXR": "3", "getErrorApox": "0.005", "getFunct": "x^2-7" },
    { "id": 2, "getXL": "4", "getXR": "5", "getErrorApox": "0.07", "getFunct": "x^2-17" },
    { "id": 3, "getXL": "2", "getXR": "9", "getErrorApox": "0.45", "getFunct": "x^4-86" },
    { "id": 4, "getXL": "2", "getXR": "15", "getErrorApox": "10", "getFunct": "x^2-79" },
    { "id": 5, "getXL": "15", "getXR": "8", "getErrorApox": "1", "getFunct": "x^2-15" }
  ],
  "FalsePositionExample": [
    { "id": 0, "getXL": "0", "getXR": "0", "getErrorApox": "0.0001", "getFunct": "0" },
    { "id": 1, "getXL": "2", "getXR": "3", "getErrorApox": "0.005", "getFunct": "x^2-7" },
    { "id": 2, "getXL": "4", "getXR": "5", "getErrorApox": "0.07", "getFunct": "x^2-17" },
    { "id": 3, "getXL": "2", "getXR": "9", "getErrorApox": "0.45", "getFunct": "x^4-86" },
    { "id": 4, "getXL": "2", "getXR": "15", "getErrorApox": "10", "getFunct": "x^2-79" },
    { "id": 5, "getXL": "15", "getXR": "8", "getErrorApox": "1", "getFunct": "x^2-15" }
  ],
  "Regression_Size": [
    { "id": 0, "getSize": "0", "getDegree": "0", "getXi": "0" },
    { "id": 1, "getSize": "4", "getDegree": "1", "getXi": "11" },
    { "id": 2, "getSize": "6", "getDegree": "1", "getXi": "18" },
    { "id": 3, "getSize": "7", "getDegree": "2", "getXi": "23" },
    { "id": 4, "getSize": "7", "getDegree": "2", "getXi": "68" },
    { "id": 5, "getSize": "8", "getDegree": "3", "getXi": "90" }
  ]
}
```

จากนั้นทำการ Run json-server ด้วยคำสั่ง “json-server --watch db.json --port 3001”

เพื่อทำการเปิด Server ที่ port 3001

```
PS C:\Users\ASUS\Desktop\Documents\Numerical_Project\my-app> json-server --watch db.json --port 3001
```

```
\{^_<sup>^</sup>}/ hi!
```

```
Loading db.json
```

```
Done
```

Resources

```
http://localhost:3001/BisectionExample
```

```
http://localhost:3001/FalsePositionExample
```

```
http://localhost:3001/Regression_Size
```

```
http://localhost:3001/RegressionExample
```

Home

```
http://localhost:3001
```

```
Type s + enter at any time to create a snapshot of the database
```

```
Watching...
```

```
GET /BisectionExample 304 28.289 ms - -
```

(ในส่วนนี้ผมทำ api ไปแค่ส่วนของ Bisection,FalsePosition และ Regression ที่เป็น Linear กับ Polynomial)

```
Numerical Method x (119) Aoi - King Atlantis | C x |
localhost:3001/BisectionExample
YouTube Maps Gmail Facebook Translate
[
  {
    "id": 0,
    "getXL": "0",
    "getXR": "0",
    "getErrorApox": "0.0001",
    "getFunc": "0"
  },
  {
    "id": 1,
    "getXL": "2",
    "getXR": "3",
    "getErrorApox": "0.005",
    "getFunc": "x^2-7"
  },
  {
    "id": 2,
    "getXL": "4",
    "getXR": "5",
    "getErrorApox": "0.07",
    "getFunc": "x^2-17"
  },
  {
    "id": 3,
    "getXL": "2",
    "getXR": "9",
    "getErrorApox": "0.45",
    "getFunc": "x^4-86"
  },
  {
    "id": 4,
    "getXL": "2",
    "getXR": "15",
    "getErrorApox": "10",
    "getFunc": "x^2-79"
  },
  {
    "id": 5,
    "getXL": "15",
    "getXR": "8",
    "getErrorApox": "1",
    "getFunc": "x^2-15"
  }
]
```

ถ้าทำการกดไปที่ลิงจะแสดงข้อมูลเป็น Array Object เก็บค่า

แต่ละตัวไว้

ทำการสร้าง Itembox ไว้สำหรับเลือกโจทย์

```
<label htmlFor='example'>&nbsp;example :&nbsp;</label>
<select name="example" id="example" onChange={getexam}>
  <option disabled selected value="0">Select โจทย์</option>
  <option value="1">ตัวอย่าง 1</option>
  <option value="2">ตัวอย่าง 2</option>
  <option value="3">ตัวอย่าง 3</option>
  <option value="4">ตัวอย่าง 4</option>
  <option value="5">ตัวอย่าง 5</option>
</select>
</div>
```

Bisection Method

XL : XR : Error approximation :

Function :

example :

ถ้าหากว่ามีการกดปุ่ม ยกตัวอย่างเช่น “ตัวอย่าง 3” จะให้ set ค่า element นี้เป็น 3 แล้วเรียกใช้ function

Getexam()

```

62 //fetch data from api
63 var getexam = e => {
64     e.preventDefault();
65     //get index
66     var d = document.getElementById("example")
67     value = d.value;
68     textt = d.options[d.selectedIndex].text;
69     console.log(value)
70     console.log(textt)

```

ทำการดึงค่าจาก element example (ที่เป็น Itembox) แล้ว set ค่าตัวแปร Global เอาไว้

3

[Bisection.js:69](#)

ตัวอย่าง 3

[Bisection.js:70](#)

จากนั้นทำการ fetch ค่าจาก api

```
77     if(value!=0) //if option is select get data from api
78     {
79         fetch('http://localhost:3001/BisectionExample') //
80         .then(res => {
81             console.log(res)
82             return res.json(); //check respond
83         })
84         .then(data => {
85             console.log(data) //show db.json
86             console.log(data[value]) // console.log for shit
87             console.log(data[value].getXR) // console.log for shit
88             setXL(data[value].getXL) //call function SetXL() with parameter of "value that store in json file"
89             setXR(data[value].getXR)
90             setErrorApox(data[value].getErrorApox)
91             setFuncnt(data[value].getFuncnt)
92         })
93         .catch(err => console.log(err))
94     }
95     getValue(); //อันนี้คือไปรัน function คำนวณหลักละ
96 }
```

ถ้ามีการเลือกตัวเลือก (value != 0) ให้ทำการดึงข้อมูลมา

บรรทัดที่ 80-83 จะเป็นการเช็ค respond ว่ามีการตอบสนองจาก server หรือไม่

```
Bisection.js:81
Response {type: 'cors', url: 'http://localhost:3001/BisectionExample', redirected: false, status: 200, ok: true, ...}
  body: (...)
  bodyUsed: true
  headers: Headers {}
  ok: true
  redirected: false
  status: 200
  statusText: "OK"
  type: "cors"
  url: "http://localhost:3001/BisectionExample"
  [[Prototype]]: Response
```

บรรทัดที่ 84-92 จะเป็นการดึงข้อมูลออกมา

ข้อมูลที่ได้เก็บไว้จะถูกเก็บเป็น **Array** ฉะนั้นเราจะดึงค่าออกตาม **index array**

แล้วทำการเรียก **Function Set**ค่า โดยนำค่า แต่ละตัวใน **json** มาใส่ใน **input form** (บรรทัด 88-91)

จากนั้นทำการเรียกฟังก์ชัน **getvalue** ตามปกติ (บรรทัด 95)

```
Bisection.js:85
▼ (6) [{...}, {...}, {...}, {...}, {...}, {...}] ⓘ
  ▶ 0: {id: 0, getXL: '0', getXR: '0', getErrorApox: '0.0001', getFunct: '0'}
  ▶ 1: {id: 1, getXL: '2', getXR: '3', getErrorApox: '0.005', getFunct: 'x^2-7'}
  ▶ 2: {id: 2, getXL: '4', getXR: '5', getErrorApox: '0.07', getFunct: 'x^2-17'}
  ▶ 3: {id: 3, getXL: '2', getXR: '9', getErrorApox: '0.45', getFunct: 'x^4-86'}
  ▶ 4: {id: 4, getXL: '2', getXR: '15', getErrorApox: '10', getFunct: 'x^2-79'}
  ▶ 5: {id: 5, getXL: '15', getXR: '8', getErrorApox: '1', getFunct: 'x^2-15'}
    length: 6
  ▶ [[Prototype]]: Array(0)

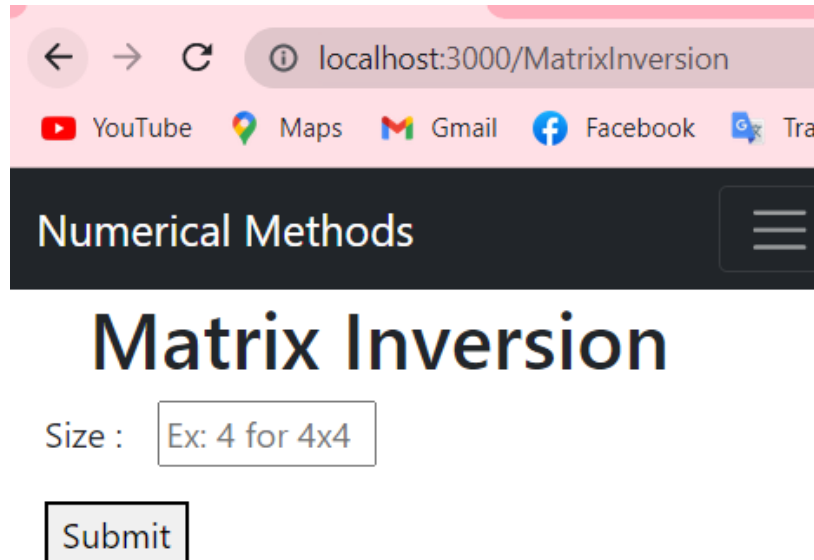
Bisection.js:86
▼ {id: 3, getXL: '2', getXR: '9', getErrorApox: '0.45', getFunct: 'x^4-86'} ⓘ
  getErrorApox: "0.45"
  getFunct: "x^4-86"
  getXL: "2"
  getXR: "9"
  id: 3
  ▶ [[Prototype]]: Object

9 Bisection.js:87
```

//หมายเหตุ **False position, One-point Iteration, Newton-Raphson, Secant method** การทำงานจะคล้ายๆกัน มีเพียงแค่ **Function** ที่ใช้ในการคำนวณมีการเปลี่ยนวิธีคิดนิดหน่อย

LINEAR ALGEBRAIC EQUATION

การคำนวณด้วยวิธี Lagrange Interpolation



← → ↻ ⓘ localhost:3000/MatrixInversion

YouTube Maps Gmail Facebook Tra

Numerical Methods

Matrix Inversion

Size :

เริ่มมาจะทำการสร้างหน้าที่รับค่า **Size** โดนถ้ามีการกดปุ่มจะทำการเรียกฟังก์ชัน `getValue`

```
153     return(<body>
154         <div>
155             <form onSubmit={getValue}>
156                 <div>
157                     <h1>&nbsp;Matrix Inversion&nbsp;</h1>
158                     <label htmlFor='Size'>&nbsp;Size :&nbsp;</label>
159                     <input
160                         name='Size'
161                         placeholder='Ex: 4 for 4x4'
162                         value = {getSize}
163                         onChange={event => setSize(event.target.value)}
164                         size='8'
165                     />
166                 </div>
167                 <p></p>
168                 <p>
169                     <div>
170                         &nbsp;<button>Submit</button>
171                     </div>
172                 </p>
173                 <p id = 'matrix'></p>
174                 <p id = 'cal_button'></p>
175                 <p id = 'outputarray'></p>
176                 <p id = 'invoutput'></p>
177                 <h3><p id = 'final'></p></h3>
178             </form>
179         </div>
180     </body>
181 )
182 }
```

```

21     var getValue = e => { //handle input event and pass value to function
22         e.preventDefault();
23         document.getElementById('matrix').innerHTML = "" //clear matrix each clicks
24         Size = getSize
25
26         console.log(Size)
27         createMatrix(Size)
28     }

```

โดยฟังก์ชัน `getValue` จะรับค่า `Size` แล้วเรียก function `createMatrix` ตามขนาดที่รับมา

```

30     function createMatrix(Size)
31     {
32         for(var row=1;row<=Size;row++)
33         {
34             for(var col=0;col<=Size;col++)
35             {
36                 document.getElementById('matrix').innerHTML += '<input type="text" id="matrix_index_row'+row+'col'+(col+1)+'" name=""
37             }
38             document.getElementById('matrix').innerHTML += '<br/>'
39         }
40         document.getElementById('cal_button').innerHTML = "" //create button
41         document.getElementById('cal_button').innerHTML += '<button onclick="calculate()">Calculate the matrix</button>'
42         document.getElementById('cal_button').onclick = function(){calculate()}; //button call calculate function
43     }
44

```

ฟังก์ชัน `createMatrix` จะทำการสร้าง `matrix` ขนาด $n \times n+1$ ออกมา

และทำการสร้างปุ่มมาอีกปุ่ม ซึ่งจะเอาผลของ `Matrix` ไปคำนวณ

(สมมุติถ้าใส่ขนาด = 4 จะสร้าง matrix 4*5 ออกมา)

Matrix Inversion

Size :

Submit

---	---	---	---	---
---	---	---	---	---
---	---	---	---	---
---	---	---	---	---

Calculate the matrix

จากนั้นถ้าเราใส่ค่าใน Matrix ทุกตัวแล้วกดคำนวณ

A				
A	A	A	A	B
A	A	A	A	B
A	A	A	A	B
A	A	A	A	B
				B

Matrix Inversion

Size :

Submit

2	35	6	89	9
74	5	6	5	3
1	2	6	8	9
1	2	6	5	88

Calculate the matrix

เมื่อกด Calculate the matrix จะทำการเรียก Function calculate()

```

45 function calculate()
46 {
47     //clear output array
48     document.getElementById('outputarray').innerHTML = ""
49
50     //get array input
51     for(var row=1;row<=Size;row++)
52     {
53         for(var col=0;col<=Size;col++)
54         {
55             var getvalue = parseFloat(document.getElementById('matrix_index_row'+row+'col'+(col+1)).value)
56             temparray.push(getvalue) //get input from form then push to array
57             console.log(temparray)
58         }
59         console.log("-----")
60         console.log(temparray)
61         array.push(temparray) //push each row to main array
62         temparray = []; // clear small array
63     }

```

ในบรรทัดที่ 55 จะเป็นการรับค่ามาเก็บใน array โดยวนรอบ element id ของ matrix

```

75 MatrixInversionCal();
76 showoutput();

```

เสร็จแล้วจะทำการเรียก Function MatrixInversionCal() และ showoutput ตามลำดับ

```

88  function MatrixInversionCal()
89  {
90      //get array value
91      var temparrayCalc = [];
92      var ending_index = parseInt(Size)+1 //get index of last element from each row
93      for(var row=1;row<=Size;row++)
94      {
95          //console.log(ending_index)
96          var temp3 = document.getElementById('matrix_index_row'+(row)+'col'+(ending_index)).value //get-push for matrix B
97          matrix_b.push(temp3)
98          for(var col=0;col<Size;col++)
99          {
100              console.log("row = "+(row)+"col = "+(col+1));
101              var temp2 = document.getElementById('matrix_index_row'+(row)+'col'+(col+1)).value //get-push matrix A
102              temparrayCalc.push(temp2)
103          }
104          matrix_A.push(temparrayCalc)
105          temparrayCalc = [];
106      }
107      //inverse of matrix A
108      matrix_A_inverse = math.inv(matrix_A);
109
110      console.log(matrix_A);
111      console.log(matrix_A_inverse);
112      console.log(matrix_b);
113
114      answerarray = math.multiply(matrix_A_inverse,matrix_b)
115      console.log(answerarray)
116  }
117

```

ในบรรทัดที่ 93-105 จะเป็นแยกค่าจาก array ใหญ่ที่ได้เป็น 2 Array ย่อยคือ A และ B

จากนั้นเปลี่ยนจาก Matrix A ให้เป็น Matrix A-1 โดยผ่าน Function ของ Library Math.js ชื่อ inv (บรรทัดที่ 108)

และทำการนำ Matrix A-1 คูณกับ Matrix B เพื่อได้ ค่า X ของแต่ละตัวออกมาด้วย multiply (บรรทัด 114)

```
function showoutput()
{
    var ans = ""//another array to store round-up value

    var invers = "==== Inversed Matrix ====<br/>"

    answerarray.forEach(arr2 => result.push(arr2.toFixed(6)))
    var times2 = 0; //index array resultinv
    console.log(result)
    for(var times=0;times<Size;times++)
    {
        matrix_A_inverse[times].forEach(arr => resultinv.push(arr.toFixed(6))) //convert all array element to 6 digit
        ans += "a("+times+1)+ " = "+result[times]+"<br/>"
    }
}
```

สุดท้ายที่ Function showoutput จะวนตาม array คำตอบที่ได้ แสดงออกมานอกหน้าจอ

Matrix Inversion

Size:

2	35	6	89	9
74	5	6	5	3
1	2	6	8	9
1	2	6	5	88

[2a(1) 35a(2) 6a(3) 89a(4) = 9]
 [74a(1) 5a(2) 6a(3) 5a(4) = 3]
 [1a(1) 2a(2) 6a(3) 8a(4) = 9]
 [1a(1) 2a(2) 6a(3) 5a(4) = 88]

==== Inversed Matrix ====

	-0.001247 0.013716 0.034913 -0.047382	
	0.030341 -0.000416 -0.849543 0.819618	
	-0.009906 -0.002147 -0.000416 0.179135	
	-0.000000 0.000000 0.333333 -0.333333	

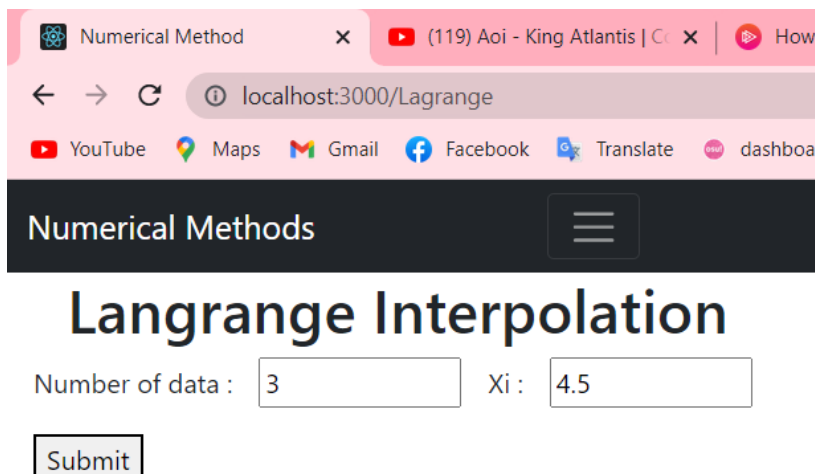
a(1) = -3.825436
 a(2) = 64.752286
 a(3) = 15.664589
 a(4) = -26.333333

INTERPOLATION

การคำนวณด้วยวิธี Lagrange Interpolation

```
129  return(<body>
130      <div>
131          <form onSubmit={getValue}>
132              <div>
133                  <h1>&nbsp;Lagrange Interpolation&nbsp;</h1>
134                  <label htmlFor='Size'>&nbsp;Number of data :&nbsp;</label>
135                  <input
136                      name='Size'
137                      placeholder='Number of size.'
138                      value = {getSize}
139                      onChange={event => setSize(event.target.value)}
140                      size='11'
141                  />
142                  <label htmlFor='Xi'>&nbsp;Xi :&nbsp;</label>
143                  <input
144                      name='Xi'
145                      placeholder='Ex: 3.5 for f(3.5)'
146                      value = {getXi}
147                      onChange={event => setXi(event.target.value)}
148                      size='11'
149                  />
150              </div>
151              <p></p>
152              <p>
153                  <div>
154                      &nbsp;<button>Submit</button>
155                  </div>
```

ส่วนนี้เหมือนกับ [หน้าที่ 22-23](#) คือสร้าง input ไว้รับ Size แต่ที่เพิ่มเติมคือ **Xi** ซึ่งเป็นค่าที่ต้องการหา



Numerical Method (119) Aoi - King Atlantis | Co x | How

localhost:3000/Lagrange

YouTube Maps Gmail Facebook Translate dashboa

Numerical Methods

Langrange Interpolation

Number of data : Xi :

แต่ฟังก์ชัน CreateMatrix() จะการสร้างแค่ Matrix ขนาด $n \times 2$ เท่านั้น

```
function createMatrix(Size)
{
  for(var row=1;row<=Size;row++)
  {
    for(var col=1;col<=2;col++)
    {
      document.getElementById('matrix').innerHTML += '<input type="text" id="matrix_index_row'+row+'col'+col+'" name="" placeholder="Value" />';
    }
    document.getElementById('matrix').innerHTML += '<br/>'
  }
  document.getElementById('cal_button').innerHTML = "<button>Calculate the matrix</button>";
  document.getElementById('cal_button').onclick = function(){calculate()};
}
```

(สมมติถ้าใส่ขนาด = 3 จะสร้าง matrix 3×2 ออกมา)

Langrange Interpolation

Number of data :

3

Xi :

4.5

Submit

-	-
-	-
-	-

Calculate the matrix

X	Y

จากนั้นถ้าเราใส่ค่าใน Matrix ทุกตัวแล้วกดคำนวณ

Langrange Interpolation

Number of data :

Xi :

Submit

6	5
2	3
4	9

Calculate the matrix

เมื่อกด Calculate the matrix จะทำการเรียก Function calculate() ซึ่งจะรับค่า Matrix แล้วเก็บเป็น Array

//คล้ายกับหน้าที่ 25 รูปที่ 1 แต่เมื่อเก็บค่าเสร็จแล้วเรียกใช้ Function คำนวณต่างกัน

```
83  function LangrangeCalc()
84  {
85      console.log("LangrangeCalc")
86      console.log(array)
87
88      //langrange cal
89      for(var a=0;a<Size;a++)
90      {
91          let term = array[a][1];
92          for(var b=0;b<Size;b++)
93          {
94              if(b!=a)
95              {
96                  term = term*(Xi - array[b][0])/(array[a][0]-array[b][0])
97              }
98          }
99          // Add current term to result
100         sum = sum + term;
101         Lterm.push(term)
102     }
103     console.log("sum = "+sum)
104     document.getElementById('final').innerHTML = "f("+Xi+" ) = "+sum.toFixed(6);
105     sum = 0;
106
107 }
```

บรรทัดที่ 89-101 จะเป็นการคำนวณ ส่วนบรรทัดที่ 103-104 จะเป็นการแสดงคำตอบ

จากนั้นทำการ print คำตอบ

```
109 function printEachLterm()
110 {
111     let proofsum = 0;
112     document.getElementById('printLterm').innerHTML = "Each L(i)F(x) term <br/>"
113     for(var i = 0;i<Size;i++) //print out each term
114     {
115         document.getElementById('printLterm').innerHTML += "L"+i+f("("+i+")")+" = "+Lterm[i]+"<br/>";
116     }
117     for(var j = 0;j<Size;j++) //proof
118     {
119         document.getElementById('printLterm').innerHTML += "+ ("+Lterm[j]+") ";
120     }
121     for(var k = 0;k<Size;k++) //proof-sum
122     {
123         proofsum += Lterm[k];
124     }
125     document.getElementById('printLterm').innerHTML += "= "+proofsum;
126 }
```

ผลลัพธ์บนเว็บ

Numerical Methods

Random links

Decha Boonmapasap

Lagrange Interpolation

Number of data : Xi :

6	5
2	3
4	9

[6x = 5y]
[2x = 3y]
[4x = 9y]

f(4.5) = 8.937500

Each L(i)F(x) term
L0f(0) = 0.78125
L1f(1) = -0.28125
L2f(2) = 8.4375
+ (0.78125) + (-0.28125) + (8.4375) = 8.9375

Uncaught ReferenceError: calculate is not defined
at HTMLButtonElement.onclick (Lagrange.js:1:1)

REGRESSION

การคำนวณด้วยวิธี Linear และ Polynomial Regression

```
397  ✓    return(<body>
398  ✓        <div>
399  ✓            <form onSubmit={getValue}>
400  ✓                <div>
401  ✓                    <h1>&nbsp;Linear / Polynomial Regression&nbsp;</h1>
402  ✓                    <label htmlFor='Size'>&nbsp;Number of data :&nbsp;</label>
403  ✓                    <input
404  ✓                        name='Size'
405  ✓                        placeholder='Number of size.'
406  ✓                        value = {getSize}
407  ✓                        onChange={event => setSize(event.target.value)}
408  ✓                        size='11'
409  ✓                    />
410  ✓                    <label htmlFor='Degree'>&nbsp;Degree :&nbsp;</label>
411  ✓                    <input
412  ✓                        name='Degree'
413  ✓                        placeholder='Ex: 2 = a0+a1x+a2x^2'
414  ✓                        value = {getDegree}
415  ✓                        onChange={event => setDegree(event.target.value)}
416  ✓                        size='18'
417  ✓                    />
418  ✓                </div>
419  ✓                <p></p>
420  ✓                <label htmlFor='Xi'>&nbsp;Xi :&nbsp;</label>
421  ✓                <input
422  ✓                    name='Xi'
423  ✓                    placeholder='Ex: X that we want to know'
424  ✓                    value = {getXi}
425  ✓                    onChange={event => setXi(event.target.value)}
426  ✓                    size='3'
427  ✓                />
428  ✓            </form>
429  ✓        </div>
430  ✓    </body>
431  ✓    )
432  ✓    }
```

ในส่วนการแสดงผล input form จะเหมือนกันกับวิธีการทำ Lagrange Interpolation

มีส่วนที่เพิ่มเติมคือ ค่า Degree ที่จะเป็นตัวกำหนดว่า เป็นฟังก์ชันแบบใด

ยกตัวอย่างเช่น ถ้าใส่ 1 ➔ $f(x) = a_0 + a_1x$

2 ➔ $f(x) = a_0 + a_1x + a_2x^2$

3 ➔ $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ เป็นต้น

Linear / Polynomial Regression

Number of data :

4

Degree :

1

Xi :

8

example :

4	5
6	9
7	16
10	18

การทำงานจะสร้าง Matrix จะเหมือนกับ Lagrange Interpolation คือสร้าง Matrix $n \times 2$

(สมมติถ้าใส่จำนวนข้อมูล = 4 จะสร้าง matrix 4×2 ออกมา) ที่เก็บ ข้อมูล x (ฝั่งซ้าย) และข้อมูล y (ฝั่งขวา) และเรียก function คำนวณ แล้วเก็บค่าทุกตัวไว้ใน Array ใหญ่

```

255 function calculate()
256 {
257     showoutputarray()
258
259     //seperate X and Y
260     for(var a=0;a<Size;a++)
261     {
262         Xarray.push(array[a][0]);
263     }
264     for(var b=0;b<Size;b++)
265     {
266         Yarray.push(array[b][1]);
267     }
268     console.log(Xarray)
269     console.log(Yarray)
270
271
272     //push x for plot graph since somehow it didnt work
273
274     RegressionCalc();
275     printout();
276

```

ฟังก์ชัน Calculate จะทำการไปเรียก showoutputarray() เพื่อเก็บค่า input อยู่ในรูป array 2มิติ

```

224 function showoutputarray()
225 {
226     //clear output array
227     document.getElementById('outputarray').innerHTML = ""
228
229     //get array input
230     for(var row=1;row<=Size;row++)
231     {
232         for(var col=1;col<=2;col++)
233         {
234             var getvalue = parseFloat(document.getElementById('matrix_index_row'+row+'col'+(col)).value)
235             temparray.push(getvalue) //get input from form then push to array
236             console.log(temparray)
237         }
238         console.log(temparray)
239         array.push(temparray) //push each row to main array
240         temparray = []; // clear small array
241     }
242     console.log(array)
243     //show array as output
244     for(var i = 0;i<Size;i++)
245     {
246         document.getElementById('outputarray').innerHTML += "[ "
247         for(var j = 0;j<1;j++)
248         {
249             document.getElementById('outputarray').innerHTML += ""+array[i][0]+"x "
250             document.getElementById('outputarray').innerHTML += " = "+array[i][1]+"y] <br/>"
251         }
252     }
253 }

```

```
▼ (4) [Array(2), Array(2), Array(2), Array(2)] ⓘ Regression.js:242
  ▶ 0: (2) [4, 5]
  ▶ 1: (2) [6, 9]
  ▶ 2: (2) [7, 16]
  ▶ 3: (2) [10, 18]
    length: 4
  ▶ [[Prototype]]: Array(0)
```

จากนั้นจะทำการแยก Array ออกเป็น 2 array คือ X กับ Y (บรรทัดที่ 260-269)

```
▼ (4) [4, 6, 7, 10] ⓘ Regression.js:268
  0: 4
  1: 6
  2: 7
  3: 10
  length: 4
  ▶ [[Prototype]]: Array(0)

▼ (4) [5, 9, 16, 18] ⓘ Regression.js:269
  0: 5
  1: 9
  2: 16
  3: 18
  length: 4
  ▶ [[Prototype]]: Array(0)
```

แล้วไปเรียกฟังก์ชัน RegressionCalc() กับ printout ตามลำดับ

ฟังก์ชัน RegressionCalc()

อธิบายวิธีคิด

ถ้าเป็น Linear Regression (Degree = 1) จะสามารถในรูป Matrix ได้ดังนี้

$$\begin{matrix} A \\ \begin{bmatrix} n & \sum x \\ \sum x & \sum x^2 \end{bmatrix} \end{matrix} \begin{matrix} x \\ \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \end{matrix} = \begin{matrix} B \\ \begin{bmatrix} \sum y \\ \sum xy \end{bmatrix} \end{matrix}$$

ถ้าเป็น Polynomial Regression (Degree = 2) จะสามารถในรูป Matrix ได้ดังนี้

$$\begin{matrix} A \\ \begin{bmatrix} n & \sum x & \sum x^2 \\ \sum x & \sum x^2 & \sum x^3 \\ \sum x^2 & \sum x^3 & \sum x^4 \end{bmatrix} \end{matrix} \begin{matrix} x \\ \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \end{matrix} = \begin{matrix} B \\ \begin{bmatrix} \sum y \\ \sum xy \\ \sum x^2 y \end{bmatrix} \end{matrix}$$

ถ้าเป็น Polynomial Regression (Degree = 3) จะสามารถในรูป Matrix ได้ดังนี้

$$\begin{matrix} A \\ \begin{bmatrix} n & \sum x & \sum x^2 & \sum x^3 \\ \sum x & \sum x^2 & \sum x^3 & \sum x^4 \\ \sum x^2 & \sum x^3 & \sum x^4 & \sum x^5 \\ \sum x^3 & \sum x^4 & \sum x^5 & \sum x^6 \end{bmatrix} \end{matrix} \begin{matrix} x \\ \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \end{matrix} = \begin{matrix} B \\ \begin{bmatrix} \sum y \\ \sum xy \\ \sum x^2 y \\ \sum x^3 y \end{bmatrix} \end{matrix}$$

จากความสัมพันธ์นี้จะสังเกตได้ว่าขนาด Matrix A คือ (Degree+1 x Degree+1)

และ Matrix B คือ (Degree+1 X 1)

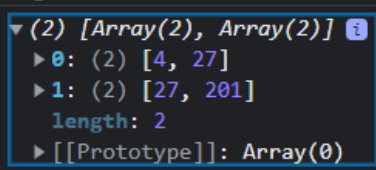
เช่นจะหา Polynomial Regression (Degree = 3)

Matrix a = 4*4 matrix b = 4*1

สร้าง Matrix A

```
function RegressionCalc()
{
  console.log("RegressionCalc")

  //Regression cal
  //get matrix A
  for(var row=0;row<=Degree;row++)
  {
    for(var col=0;col<=Degree;col++)
    {
      let mul = row+col
      tempRegression.push(SumofSquareX(Xarray,mul)); //call function to mul the sum
    }
    MatrixA.push(tempRegression) //push to array matrix A
    tempRegression = []; //clear array for next inc row
  }
  console.log(MatrixA)
}
```



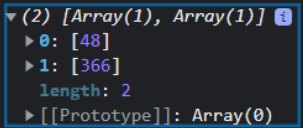
ตัวแปร mul คือ row+col สมมุติถ้า row = 1 col = 1 ให้สร้าง array อีกตัวหนึ่ง

(ในที่นี้คือ tempRegression) ที่เก็บค่า ผลรวมของ X^2 (= 201) แล้ว push ไปใน MatrixA

```
function SumofSquareX(arr,mul) //function return X^...
{
  let sumofsqrt = 0.0;
  for(var i = 0;i<Size;i++)
  {
    sumofsqrt += Math.pow(arr[i],mul);
  }
  return sumofsqrt
}
```


สร้าง Matrix B

```
310 for(var row2=0;row2<=Degree;row2++)
311 {
312     //console.log(row2)
313     tempRegression.push(SumofSquareY(Xarray,Yarray,row2)); //call function to mul the sum
314     MatrixB.push(tempRegression) //push to array matrix B
315     tempRegression = []; //clear array for next inc row
316 }
317 console.log(MatrixB)
```



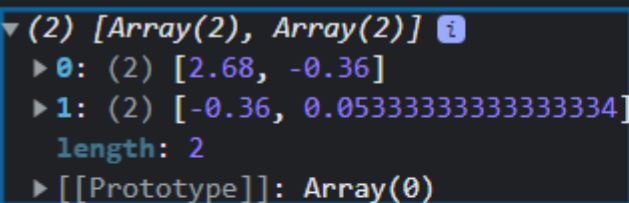
คล้ายๆกับการหา Matrix ของ A แต่จะจากที่จะเป็น ผลรวมของ $X^{\text{col}+\text{row}}$

เป็นหาผลรวมของ $X^{\text{row}} * Y$ แทน แล้ว push ใส่ Matrix B

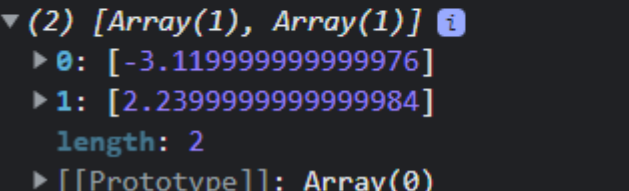
```
336 function SumofSquareY(arr,arr2,mul2) //function return  $X^{\text{row}} * Y$ 
337 {
338     let sumofsqr2 = 0.0;
339     for(var i = 0;i<Size;i++)
340     {
341         sumofsqr2 += Math.pow(arr[i],mul2)*arr2[i]
342     }
343     return sumofsqr2
344 }
```

สุดท้ายหา a ของแต่ละตัวโดยใช้การวิธี Matrix inversion

```
319 //find each a with using Matrix inverse
320 MatrixA_Inv = math.inv(MatrixA)
321 answerarray = math.multiply(MatrixA_Inv,MatrixB);
322 console.log(MatrixA_Inv)
323 console.log(answerarray)
```



Regression.js:322



Regression.js:323

และเมื่อคำนวณทุกอย่างเสร็จสิ้นจะทำการเรียกคำสั่ง `printout` เพื่อทำการแสดงข้อมูล

```
346 function printout()
347 {
348
349     document.getElementById('final').innerHTML = ""
350     document.getElementById('proof').innerHTML = ""
351
352     for(var i = 0;i<=Degree;i++)
353     {
354         document.getElementById('final').innerHTML += "a("+i+") = "+answerarray[i]+"<Br/>"
355     }//print out answer
356     console.log(Xi)
357
358     //proof if insert xi
359     if(Xi!='')
360     {
361         let sumproof = 0;
362         document.getElementById('proof').innerHTML = "g("+Xi+") ="
363         for(var j = 0;j<=Degree;j++)
364         {
365             document.getElementById('proof').innerHTML += "("+answerarray[j]+"*"+Math.pow(Xi,j)+"");
366             sumproof += answerarray[j]*Math.pow(Xi,j);
367         }
368         document.getElementById('proof').innerHTML += " =" + sumproof;
369     }
370
371     //แทนค่า ทุกตัวในสมการหา x ที่ใส่ input plot graph
372     for(var k = 0;k<Xarray.length;k++)
373     {
374         let regressionresult = 0;
375         for(var l = 0;l<=Degree;l++)
376         {
377             regressionresult += answerarray[l]*Math.pow(Xarray[k],l);
378         }
379         Y_Regressionarray.push(regressionresult)
380     }
381     console.log(Xarray)
382     console.log(Yarray)
383     console.log(Y_Regressionarray)
384     //เอาค่าใส่ array ไว้ plot graph
385     for(var d = 0;d<Y_Regressionarray.length;d++)
386     {
387         let tempa = []
388         tempa.push(Xarray[d])
389         tempa.push(Y_Regressionarray[d])
390         array2.push(tempa)
391     }
392     console.log(array2)
393
394
395 }
```