10.4.1.2.2

EE24BTECH11006 - Arnav Mahishi

Question:
The product of two consecutive integers is 306. We need to find the integers.

Variable	**Description**
x	The bigger integer out of the two we need to find
$g(x)$	The function we take to update x_n in the point iteration method
$x_{\alpha,n}$	The value of x after n iterations for the first root
$x_{\beta,n}$	The value of x after n iterations for the second root

TABLE 0: Caption

TABLE 0: Variables Used

Theoretical Solution:
Lets start by assuming the bigger integer as x and the smaller integer as $\frac{306}{x}$

$$\implies x - \frac{306}{x} = 1 \tag{0.1}$$

$$\implies x^2 - 306 = x \tag{0.2}$$

$$\implies x^2 - x - 306 = 0 \tag{0.3}$$

Using the quadratic formula:

$$x = \frac{1 \pm \sqrt{1^2 - (4 \cdot -306)}}{2} \tag{0.4}$$

$$x_1 = \frac{1 + \sqrt{1225}}{2} = 18 \tag{0.5}$$

$$x_2 = \frac{1 - \sqrt{1225}}{2} = -17 \tag{0.6}$$

If $x = 18$ the other integer will be 17 if $x = -17$ the other integer will be -18

$$p(2) = 41 - 72(-2) - 18(-2)^2 = 113 \tag{0.7}$$

$\therefore$ The integers can be $18, 17$ or $-18, -17$
Computational Solution:
Below are three methods to find the solutions of this quadratic equation,
Matrix-Based Method:

For a polynomial equation of form $x_n + b_{n-1}x^{n-1} + \cdots + b_2x^2 + b_1x + b_0 = 0$ we construct a matrix called companion matrix of form

$$\Lambda = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \vdots & 1 \\ -b_0 & -b_1 & -b_2 & \dots & -b_{n-1} \end{pmatrix} \tag{0.8}$$

The eigenvalues of this matrix are the roots of the given polynomial equation. To find the eigenvalues of the matrix we use QR decomposition with Wilkinson's shift. The QR Algorithm with Wilkinson's shift modifies the traditional QR iteration to improve the convergence rate by incorporating a shift that is carefully computed. This shift is based on the bottom right 2×2 submatrix of the matrix A and is given by:

$$\text{shift} = A_{(n)(n)} - \text{sgn}\,(d)\ \sqrt{d^2 + A_{(n)(n-1)}A_{(n-1)(n)}}, \tag{0.9}$$

where

$$d = \frac{A_{(n-1)(n-1)} - A_{(n)(n)}}{2}. \tag{0.10}$$

N is the size of the matrix. This strategy allows the algorithm to more rapidly converge to the eigenvalues of the matrix. The steps of the Algorithm are as follows:

1) **Input:**
   - A square matrix A of size $n \times n$.
   - Maximum number of iterations, Max_iterations (default: 1000).
   - Convergence tolerance, Tolerance (default: 10^{-10}).
2) **Initialization:**
   - Set $A_0 = A$, the input matrix.
   - Initialize the iteration counter, iter = 0.
3) **Iterative Process:** Repeat the following steps until the matrix converges (sub-diagonal elements are below Tolerance) or iter = Max_iterations:
   a) **Compute Wilkinson's Shift:** For the bottom-right 2×2 submatrix:

   $$B = \begin{bmatrix} a_{n-2,n-2} & a_{n-2,n-1} \\ a_{n-1,n-2} & a_{n-1,n-1} \end{bmatrix}, \tag{3.1}$$

   calculate:

   $$\text{shift} = A_{(n)(n)} - \text{sgn}\,(d)\ \sqrt{d^2 + A_{(n)(n-1)}A_{(n-1)(n)}}, \tag{3.2}$$

   where

   $$d = \frac{A_{(n-1)(n-1)} - A_{(n)(n)}}{2}. \tag{3.3}$$

   where sign(d) is 1 if $d \geq 0$ and -1 otherwise and N is the size of the matrix

b) **Shift the Matrix:** Subtract μ from the diagonal elements of A_k:

$$\hat{A}_k = A_k - \mu I. \tag{3.4}$$

c) **QR Decomposition:** Decompose $\hat{A}_k$ into Q_k (orthogonal) and R_k (upper triangular) using Gram-Schmidt:

$$\hat{A}_k = Q_k R_k. \tag{3.5}$$

d) **Update the Matrix:** Compute the next matrix:

$$A_{k+1} = R_k Q_k + \mu I. \tag{3.6}$$

e) **Check Convergence:** If all sub-diagonal elements $\hat{A}_k[i+1, i]$ are smaller than Tolerance, break the iteration loop.

4) **Extract Eigenvalues:** For the resulting matrix, if any 2×2 submatrix remains on the diagonal, compute its eigenvalues using the quadratic equation:

$$\lambda^2 + b\lambda + c = 0, \tag{4.1}$$

where $b = -(a_{n-2,n-2} + a_{n-1,n-1})$ and $c = a_{n-2,n-2}a_{n-1,n-1} - a_{n-1,n-2}a_{n-2,n-1}$. For diagonal elements, take $\lambda = a_{i,i}$.

5) **Output:** The eigenvalues of the matrix A.

In our case:

$$\Lambda = \begin{pmatrix} 0 & 1 \\ 306 & 1 \end{pmatrix} \tag{5.1}$$

The eigen values given by the code are

$$x_1 = 18.000000000433293 \tag{5.2}$$

$$x_2 = -17.000000000043303 \tag{5.3}$$

Theoritically we can solve it using:

$$|\Lambda - \lambda I| = 0 \tag{5.4}$$

$$\implies \left| \begin{pmatrix} 0 & 1 \\ 306 & 1 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right| = 0 \tag{5.5}$$

$$\implies \left| \begin{pmatrix} -\lambda & 1 \\ 306 & 1-\lambda \end{pmatrix} \right| = 0 \tag{5.6}$$

$$\implies \lambda^2 - \lambda - 306 = 0 \tag{5.7}$$

$$\lambda = 18, -17 \tag{5.8}$$

$\therefore$ The roots of the equation are 18 and -17

Fixed Point Iterations:

Take an initial guess x_0. The update difference equation will use the following function:

$$x = g(x) \tag{5.9}$$

For our problem,

$$g(x) = \sqrt{x^2 - 306} \tag{5.10}$$

To get both roots we do two iterations, now the update equations will be

$$x_{\alpha,n+1} = g(x_{\alpha,n}) \tag{5.11}$$
$$x_{\beta,n+1} = g\left(x_{\beta,n}\right) \tag{5.12}$$
$$\tag{5.13}$$

We take two iniital guesses $x_{\alpha,0}$ and $x_{\beta,0}$ close to each root. Then we continue calculating the each $x_{\alpha,n}$ and $x_{\beta,n}$ until

$$|x_{n+1} - x_n| < \epsilon \tag{5.14}$$

Where ϵ is the tolerance which we have taken as 1e-6. In each of the α series and β series we get a root.

This is proved by a theorem as follows:
Let $x = s$ be a solution of $x = g(x)$ and suppose that g has a continuous derivative in some interval J containing s. Then if $|g'| \leq K < 1$ in J, the iteration process defined above converges for any x_0 in J. The limit of the sequence $[x_n]$ is s.

This can also be solved by the Newton-Raphson Method,
Start with an initial guess x_0, and then run the following logical loop,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{5.15}$$

where ,

$$f(x) = x^2 - x - 306 \tag{5.16}$$
$$f'(x) = 2x - 1 \tag{5.17}$$

The problem with this method is if the roots are complex but the coeffcients are real, x_n either converges to an extrema or grows continuously without any bound. To get the complex solutions, however , we can just take the initial guess point to be a random complex number.
The output of a program written to find roots is shown below:

```
Fixed-Point Iteration for Positive Root:
Iteration 1: x = 17.577532, f(x) = -14.607907
Iteration 2: x = 17.988261, f(x) = -0.410729
Iteration 3: x = 17.999674, f(x) = -0.011413
Iteration 4: x = 17.999996, f(x) = -0.000237
Iteration 5: x = 18.000000, f(x) = -0.000009
Converged to solution: x = 18.000000
```

```
Fixed-Point Iteration for Negative Root:
Iteration 1: x = -9.767519, f(x) = -200.828057
Iteration 2: x = -17.211406, f(x) = 7.443887
Iteration 3: x = -16.993781, f(x) = 0.217625
Iteration 4: x = -17.001183, f(x) = -0.006402
Iteration 5: x = -16.999995, f(x) = 0.000018
Iteration 6: x = -17.000000, f(x) = -0.000000
Converged to solution: x = -17.000000

Newton-Raphson Iteration for Positive Root:
Iteration 1: x = 19.448724, f(x) = 52.804159
Iteration 2: x = 18.055381, f(x) = 1.941406
Iteration 3: x = 18.000887, f(x) = 0.003057
Iteration 4: x = 18.000001, f(x) = 0.000007
Iteration 5: x = 18.000000, f(x) = 0.000000
Converged to solution: x = 18.000000

Newton-Raphson Iteration for Negative Root:
Iteration 1: x = -19.809749, f(x) = 106.235914
Iteration 2: x = -17.194357, f(x) = 6.840276
Iteration 3: x = -17.007687, f(x) = 0.073361
Iteration 4: x = -17.000017, f(x) = 0.000307
Iteration 5: x = -17.000000, f(x) = 0.000000
Converged to solution: x = -17.000000

Eigenvalues (Roots of the equation):
Eigenvalue 1: -17.000000 +0.0000000i
Eigenvalue 2: 18.000000 + 0.0000000i
```