Numerical Analysis on Root finding for Nonlinear Eqn's

(A) Motivation

In thermal dynamics or fluid mechanics, engineers often encounter complicated functions that need to be solved.

For example, to model a wind driven air flow through a room using conservation of mass, I would need to solve an eqn like the following:

\[ \mathcal{K}_i \int \frac{2E(P_{out}) - P_{wind,i} - P_{in}}{\rho} \, dl - \mathcal{K}_o \int \frac{2E(P_{in}) - (P_{out} + P_{wind,o})}{\rho} \, dl = 0. \]

In here, we would need to solve for \( P_{out} \). But there's no good way to do so by hand. In cases like this, we need to use numerical method, which discretizes continuous functions for computation.

Today, I'm going to talk about some root finding methods that are commonly used and also how fast they converge. Numerical analysis.

Side board:

Three common methods

<table>
<thead>
<tr>
<th>Method</th>
<th>PRO's</th>
<th>CON's</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Bisection</td>
<td>can always find a root</td>
<td>slow</td>
</tr>
<tr>
<td>2. Newton-Raphson</td>
<td>might fail to find a root</td>
<td>fast</td>
</tr>
<tr>
<td>3. Secant</td>
<td>might fail to find a root</td>
<td>don't need ( f'(x) )</td>
</tr>
</tbody>
</table>
Given $f(x), f'(x)$ at any $x$.

- You want to find $x^*$.
- You look between interval $[a_0, b_0]$ where $a_0 \cdot b_0 < 0$, meaning one of the values has to be (-) and the other ( ), and $a_k < x^* < b_k$.
- In this case $f(a_k) > 0, f(b_k) < 0$, where $k = \# \text{iteration}$.
- We also want to look for their midpoint $m_k = \frac{1}{2} (a_k + b_k)$.

3 cases:

- If $f(m_k) < 0$, then
  
  $a_{k+1} = a_k$
  
  $b_{k+1} = m_k$.

- If $f(m_k) > 0$,
  
  $a_{k+1} = m_k$
  
  $b_{k+1} = b_k$.

- If $f(m_k) = 0$, we've found our solution.

- Repeat until $|f(m_k) - 0| < \varepsilon$.

(Note: $\varepsilon$ is an arbitrary # that you set.)
• Speed of convergence

\[ |x^* - a_k| \leq |b_k - a_k| = \frac{1}{2^k} |b_0 - a_0| \]

distance between \( x^* \) and \( a_k \)

converge @

\[ |x^* - a_k| = |\varepsilon_k| \leq \frac{1}{2^k} |b_0 - a_0| \rightarrow \text{get a binary digit of accuracy each time} \]

\[ \uparrow \text{error gets halved} \]

@ each iteration.

• demo w/MATLAB

\[ f(x) = x^2 - 2 \]

0 = \( x^2 - 2 \) \( \Rightarrow \sqrt{2} \approx 1.41421356237 \)

⇒ Run code to get five digits of accuracy

takes 16 iterations

\[ \text{slow} \]

side board

PRO: can always find a root

CON: slow.
2. **Newton-Raphson Method.**

Given $f(x)$ and $f'(x)$ at any $x$.

![Graph of $f(x)$ and tangent lines](image)

- Start at $x_0$.
- Find $f'(x_0)$.
- Project $f'(x_0)$ tangent line and see where it hits $f(x) = 0$.
- That's your new starting point.
- Projected for the tangent line:
  
  $y = f(x_0) + f'(x_0)(x - x_0)$

- Want it to hit $0$;

  $y = 0$

- Combine:
  
  $0 = f(x_0) + f'(x_0)(x - x_0)$

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

- Demo using MATLAB:
  
  Converge to at least 15 digit of accuracy by 6th iteration.
  
  The # of correct digits get doubled every iteration.

- Cases when it fails:
  
  ![Slope of $\tan(x)$ and $f(x)$](image)
Let $E_n = X_n - X^*$

Assume $f''$ continuous

$|f''(x)| > 0$ in neighborhood of $x^*$

Use Taylor expansion about $X_n$:

$\frac{f(x^*) - f(X_n)}{0} = f(X_n) + f'(X_n)(x^* - X_n) + \frac{f''(\xi)}{2} (x^* - X_n)^2 + \ldots$

$\Rightarrow 0 = f'(X_n) (x^* - X_{n+1})$ ← Newton's method

$\frac{f''(\xi)}{2} (x^* - X_{n+1}) = \frac{-1}{2} (x^* - X_n)^2 f''(\xi)$

$E_{n+1} = \frac{-f''(\xi)}{2 f'(X_n)} E_n^2$

In limit $n \to \infty$, $\frac{-f''(\xi)}{2 f'(X_n)} \to \frac{-f''(x^*)}{2 f'(x^*)} \to \text{constant}$

$\Rightarrow |E_{n+1}| \leq C E_n^2$ ← Quadratic convergence.

= double # of correct digits every time

Ex. $E_n = 10^{-5}$ (5 digits accuracy)

$E_{n+1} \approx 10^{-10}$ (10 digits accuracy)
So:\[
\varepsilon_0 \\
\varepsilon_1 \sim \frac{\varepsilon_0^2}{C} \\
\varepsilon_2 \sim \frac{\varepsilon_0^4}{C^2} \\
\varepsilon_3 \sim \frac{\varepsilon_0^8}{C^3} \\
\vdots \\
\varepsilon_k \sim \frac{\varepsilon_0^{2k-1}}{C^{2k}} = \frac{1}{C} (C\varepsilon_0)^{2k}.
\]

If we want convergence, \(\varepsilon_k \to 0\) (not blow up) we need \(|C\varepsilon_0| < 1\).

Recall:
\[
C = \frac{-f''(x)}{2f'(x)}
\]

\[
\left| \frac{-1}{2} \frac{f''(x)}{f'(x)} \varepsilon_0 \right| < 1
\]

To consider worst case:
\[
\frac{1}{2} \max_{y \in N} \left| \frac{f''(y)}{f'(y)} \right| |\varepsilon_0| < 1
\]

\(\max\) possible value.

For \(\arctan\), since \(f'(x)\) is very small, iteration often blows up unless very close to \(x^*\) (\(|\varepsilon_0|\) small).

**PRO:** very fast

**CON:** may fail to find root unless you're quite close to it.
(3) **Secant Method**

Given \( f(x) \).

Similar to Newton's method, except that you don't need \( f'(x) \).

Replace \( f'(x) \) with:

\[
f'(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} = f(x_{n-1}, x_n)
\]

Similar to finite difference

\[x_{n+1} = x_n - \frac{f(x_n)}{f(x_{n-1}, x_n)}\]

- Demo using MATLAB.
  - Reaches 5 digit accuracy after 6 iterations.

- **Convergence**
  - Let \( \varepsilon_n = x_n - x^* \)
  - Polynomial of order 1:
    \[p(x) = f_n + f[x_{n-1}, x_n](x - x_{n-1})\]
using Taylor Expansion:

\[
f(x^*) = f(x_n) + f'[x_{n-1}, x_n] (x^* - x_n) + \frac{1}{2} f''(\xi) (x^* - x_n)(x^* - x_{n-1})
\]

\[\xi \in (x^*, x_n)\]

\[
0 = f(x_n) + f'[x_{n-1}, x_n] (x_{n+1} - x_n)
\]

\[
0 = f[x_{n-1}, x_n] (x^* - x_{n+1}) + \frac{1}{2} f''(\xi) (x^* - x_n)(x^* - x_{n-1})
\]

\[\xi \in E_{n+1}\]

\[E_{n+1} = \frac{1}{2} \frac{f''(\xi)}{f'[x_{n-1}, x_n]} E_n E_{n-1}\]

as \( n \to \infty \)

bounded locally \( f''/f' \) turns to \( C \)

\[|E_{n+1}| \leq C |E_n| |E_{n-1}|\]

To put into a more precise form:

Let \( |E_n| \leq C |E_{n-1}|^p \) to be determined accordingly:

\[|E_{n+1}| \leq C |E_n|^p \leq C |E_{n-1}|^p^2\]

together:

\[|E_{n-1}|^p^2 \leq C |E_{n-1}|^p |E_{n-1}|\]

matching the exponents on both sides:

\[p^2 = p + 1\]

\[p = \frac{1 + \sqrt{5}}{2} \approx 1.618, \text{ golden ratio}\]

compared to the quadratic convergence of

Newton's method + the linear convergence of

Bisection, secant method falls in blow them.

PRO: don't need to know \( f(x) \)

CON: may not converge

need to be close to root.