CS3430 S26: Scientific Computing
My Observations on HW 5, Problem 2 – Leibniz and Machin

Vladimir Kulyukin

Problem 2: Interpretation of Unit Test Output

In this problem, I compared two generator-based algorithms for approximating π: the Leibniz series and Machin's formula. Both implementations use `mpmath` with the same working precision, but they differ radically in convergence speed.

Structural Expectations

The structural tests confirm that both required generators exist and are callable:

- `pi_leibniz_mp`
- `pi_machin_mp`

They also confirm that both generators yield multiple successive approximations of π. The first three values produced by each generator were:

```
Leibniz first 3 approximations:
[4.0, 2.666666..., 3.466666...]

Machin first 3 approximations:
[3.18326..., 3.14059..., 3.14162...]
```

Even at this early stage, the difference is visible: the Leibniz series oscillates widely, while Machin's formula is already close to π.

Convergence Behavior: Leibniz vs. Machin

The convergence test evaluates both methods using the same precision (`mp.dps = 60`) but different iteration counts.

Leibniz (2000 terms). After 2000 terms, the Leibniz generator produced:

```
Approximation: 3.1410926536210432...
Absolute error: ~ 5.0 x 10^{-4}
```

This confirms the lecture discussion: the Leibniz series converges linearly, and each additional decimal digit requires roughly an order of magnitude more terms. Even thousands of iterations barely produce three correct decimal places.

Machin (10 terms). After only 10 terms, the Machin generator produced:

```
Approximation: 3.1415926535897916...
Absolute error: ~ 1.5 x 10^{-15}
```

This is essentially double-precision accuracy, achieved with the same arctangent series used by Leibniz, but evaluated at much smaller arguments.

Key Lesson Reinforced by the Tests

The unit tests demonstrate that:

- both algorithms are mathematically correct,
- both rely on the same Taylor expansion of $\arctan(x)$,
- but their computational performance differs by orders of magnitude.

In concrete terms:

Method	**Iterations**	**Error**
Leibniz	2000	$\sim 10^{-4}$
Machin	10	$\sim 10^{-15}$

This difference reflects an algorithmic transformation, not a constant-factor speedup.

Takeaway

Problem 2 illustrates a central principle of scientific computing: correctness alone is insufficient. Convergence rate and problem formulation dominate practical performance. Scientific computing rewards transforming a problem into a numerically favorable form rather than relying on brute force.