ENGSCI 233 - Numerical Errors
Lecture 2

Colin Simpson

Semester One 2022

Department of Engineering Science
University of Auckland
(Beamer Theme: Metropolis)

Truncation Error

Leibniz Series and Truncation Error

Consider the Leibniz series representation of π:

$$\pi = 4 \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

However, we can't do infinite sums in reality. Instead, we must truncate the series at N terms:

$$\pi \approx 4 \sum_{k=0}^{N} \frac{(-1)^k}{2k+1}$$

The difference is the *truncation error*. It is a property of the method or algorithm, rather than of floating point precision.

Truncation Error Example

The Leibniz series returned the same result in double, single and half precision. However, they are clearly not equal to π.

We therefore are encountering an issue with truncation error. It is a property of the Leibniz series, rather than floating point precision.

Handling Truncation Error

We have a few ways of dealing with truncation error. For example:

- Increase the number of terms included in the series. However, this takes more time and may increase the risks of floating point errors (e.g. representation, rounding, accumulation).
- Use a different algorithm. This may allow us to arrive at an answer faster or with greater accuracy.

Numerical Convergence Tests

Convergence Testing

How do we know when the result of an iterative algorithm is close to the true result?

If we are able to find the true result by other means, we can use this to compare with the result from our algorithm.

Failing that, we can investigate the effect from increasing the number of iterations. This is known as *convergence testing*.

Convergence of Iterative Algorithms

Ideally, the result of an iterative algorithm will continually approach the true result as the number of iterations increases.

In other words, the truncation error will gradually decrease.

If we don't know the true result, then we don't know the true truncation error. However, we do know the relative change in our algorithm results as we vary the number of iterations.

Numerical Convergence Tests

There are three commonly used numerical convergence tests for a sequence of algorithm results, $\vec{x}$:

- Absolute value test:

$$|x_k - x_{k-1}| < \epsilon$$

- Relative test:

$$\frac{|x_k - x_{k-1}|}{|x_k|} < \epsilon$$

- Uniform test:

$$\frac{|x_k - x_{k-1}|}{1 + |x_k|} < \epsilon$$

where k is the iteration number and ϵ is an error tolerance.

Numerical Convergence Tests

The absolute value and relative tests can be sensitive to the size of the algorithm results, $\vec{x}$:

- Absolute value test highly sensitive to large $\vec{x}$.
- Relative test highly sensitive to small $\vec{x}$.
- Uniform test acts like absolute value test for small $\vec{x}$ and relative test for large $\vec{x}$.

We can apply a numerical convergence test in our algorithm until it reaches a desired error tolerance.

Numerical Convergence Test Example

Let's consider the example of the Leibniz series again.

As we know the exact value of π, we can compare the numerical convergence tests to the actual truncation error.

The Leibniz series is said to have *slow convergence* i.e. we must greatly increase N for the series to approach the true value of π.