# HW1 Report - Chebyshev

## Problem Definition

Given a function $f(x)$ over an interval $[a, b]$, the task is to approximate $f(x)$ using a polynomial. Traditional methods like Lagrange or Newton interpolation often suffer from oscillations or high error at the edges of the interval, known as Runge's phenomenon.
Chebyshev interpolation mitigates these issues by employing Chebyshev polynomials $T_n(x)$ and Chebyshev nodes $x_i$ .

### Chebyshev Nodes

The n+1 Chebyshev nodes, $x_i$, are the roots of $T_{n+1}(x)$ over $[-1, 1]$. These nodes cluster near the interval endpoints, resulting in a non-uniform distribution that helps mitigate Runge's phenomenon.

### Chebyshev Interpolation Formula

Given $f(x)$ and Chebyshev nodes $x_0, x_1, \ldots, x_n$, the Chebyshev interpolation polynomial $P_n(x)$ of degree n is:

$$P_n(x) = \sum_{k=0}^{n} c_k \cdot T_k(x)$$

where $c_k$'s are the interpolation coefficients obtained by solving a system of linear equations using $f(x_i)$.
For a given n, define the Chebyshev coefficients by

$$c_i = \frac{2}{n} * \sum_{j=0}^{n-1} f(x(j, n)) * T_{i-1}(x(j, n))$$

*the accurate formula is an integral, here we use the computational discrete version.*

## Algorithms

- In *chebyshev_node.m*, I compute the Chebyshev nodes and then fit a polynomial at these points.
- In *chebyshev_interpolation*, I implement the standard Chebyshev interpolation over $[a, b]$: compute $c_i$ first, then assemble the interpolation formula.

## Results

See the *result* folder, or run the programs if you have trouble accessing the *result* folder.

## Issues / Takeaways

- doing polynomial interpolation with Chebyshev nodes can be much faster than with Chebyshev functions. *but we need to use a relatively small degree* to avoid instability and oscillation, **theoretically, interpolation with nodes can be just as precise as with Chebyshev functions.**
- Chebyshev interpolation fails when we set $[a, b]$ too wide, such as $[-10, 10]$.
- Maximum error unsurprisingly goes to zero pretty quickly.