MAT128A: Numerical Analysis
Lecture Sixteen: Equispaced Interpolation Nodes

November 2, 2018

Review

We saw last time that Chebyshev nodes are "good" interpolation nodes.

More explicitly, if $f : [-1, 1] \to \mathbb{R}$ is continuous and

$$p_N(x) = \sum_{n=0}^{N}{}' a_n T_n(x) \quad \text{with} \quad a_n = \frac{2}{\pi} \int_0^\pi f(x) T_n(x) \, \frac{dx}{\sqrt{1-x^2}},$$

then

$$\|f - P_N\|_\infty \leq \mathcal{O}\left(\log(N)\right) \|f - P_N^*\|_\infty ,$$

where P_N^* is the minimax polynomial. If f is smooth enough then we have

$$\|f - P_N\|_\infty \leq \mathcal{O}\left(1\right) \|f - P_N^*\|_\infty ,$$

and this is the case for most functions of interest.

Review

What's more is that we have simple algorithms for approximating the Chebyshev expansion of a function f. For instance, if

$$\widetilde{P_N}(x) = {\sum_{n=0}^{N}}' \widetilde{a_n} T_n(x) \quad \text{with} \quad \widetilde{a_n} = \frac{2}{N+1} \sum_{j=0}^{N} f\left(\cos\left(\frac{j+\frac{1}{2}}{N}\pi\right)\right) T_n\left(\cos\left(\frac{j+\frac{1}{2}}{N}\pi\right)\right),$$

then

$$\left\| P_N(x) - \widetilde{P_N}(x) \right\|_\infty \leq \sum_{n=N+1}^{\infty} |a_n| .$$

Of course, $\widetilde{P_N}$ is the unique polynomial of degree N which interpolates f at the nodes

$$\cos\left(\frac{j+\frac{1}{2}}{N}\pi\right) \quad j = 0, 1, \ldots, N.$$

Other choices of Interpolation Nodes

So we have a rigorous statement to the effect that Chebyshev nodes are "good."

But how do they compare to other choices of interpolation nodes? As far as we know at this point, any set of interpolation nodes might be almost as good as Chebyshev nodes.

Perhaps being close to the minimax approximation is a typical property enjoyed by pretty much any collection of interpolation nodes.

Equispaced Interpolation Nodes

In this lecture, we will see that this is not the case at all.

Indeed, we will see that equispaced interpolation nodes are **bad** interpolation nodes in a rather decisive way. This is somewhat vexing since equispaced nodes are one of the most natural and obvious choices.

Note that this does not mean that one never performs interpolation using equispaced nodes. In fact, interpolation using equispaced nodes is one of the most commonly used techniques in all of numerical analysis — we will discuss why after we see that equispaced nodes are "bad."

Equispaced Interpolation Nodes

Theorem (Runge's Example)

Let $f : [-1, 1]$ *be defined by*

$$f(x) = \frac{1}{1 + 25x^2},$$

and, for each positive integer N*, let* Q_N *be the polynomial of degree* N *which interpolates* f *at the points*

$$-1 + \frac{2j}{N}, \quad j = 0, 1, \ldots, N.$$

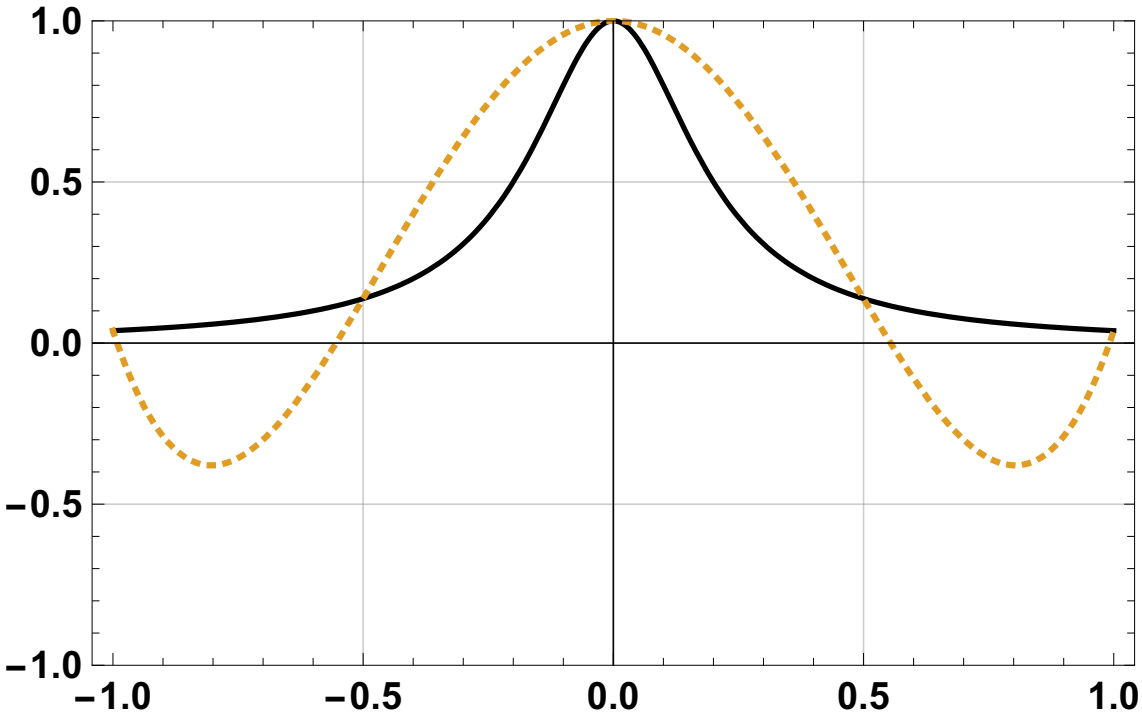
Then

$$\|f - Q_N\|_\infty = \mathcal{O}\left(2^N\right).$$

Here we see that not only does the interpolating polynomial not converge, the difference in the uniform norm grows **exponentially fast** in N (ouch).

Numerical Experiments

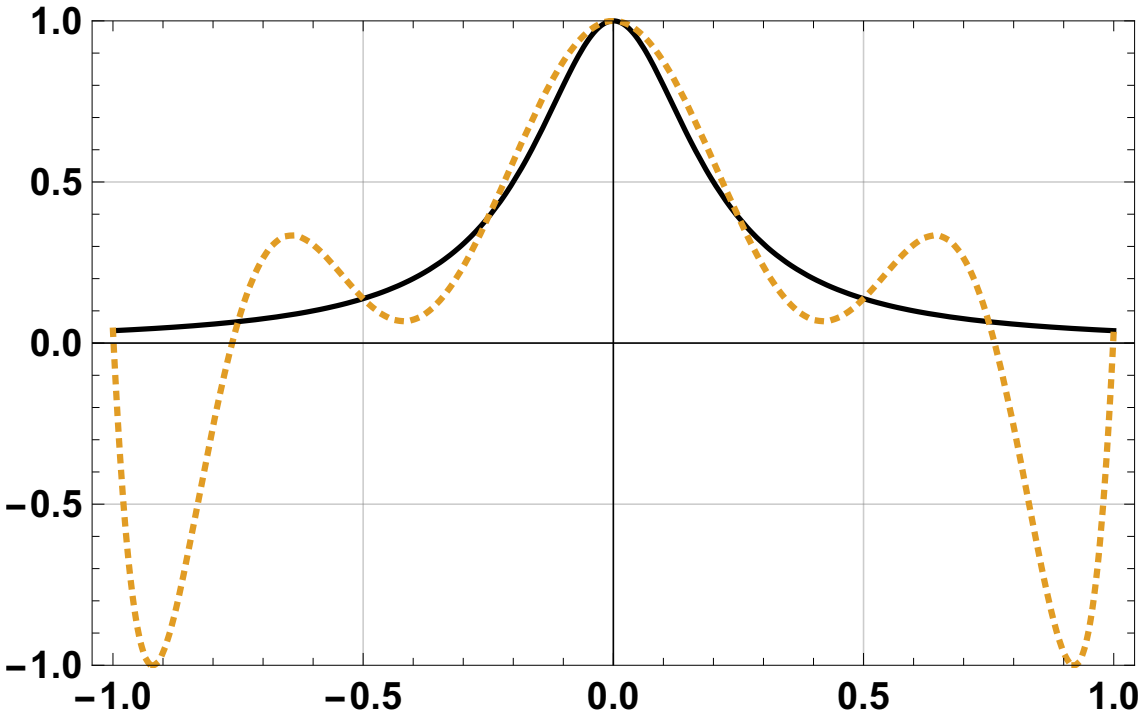
We will not prove this theorem, but we will do some numerical experiments to demonstrate the result. First, let's do some numerical experiment to test the hypothesis.



$$f(x) = \frac{1}{1 + 25x^2} \quad N = 4$$

Numerical Experiments

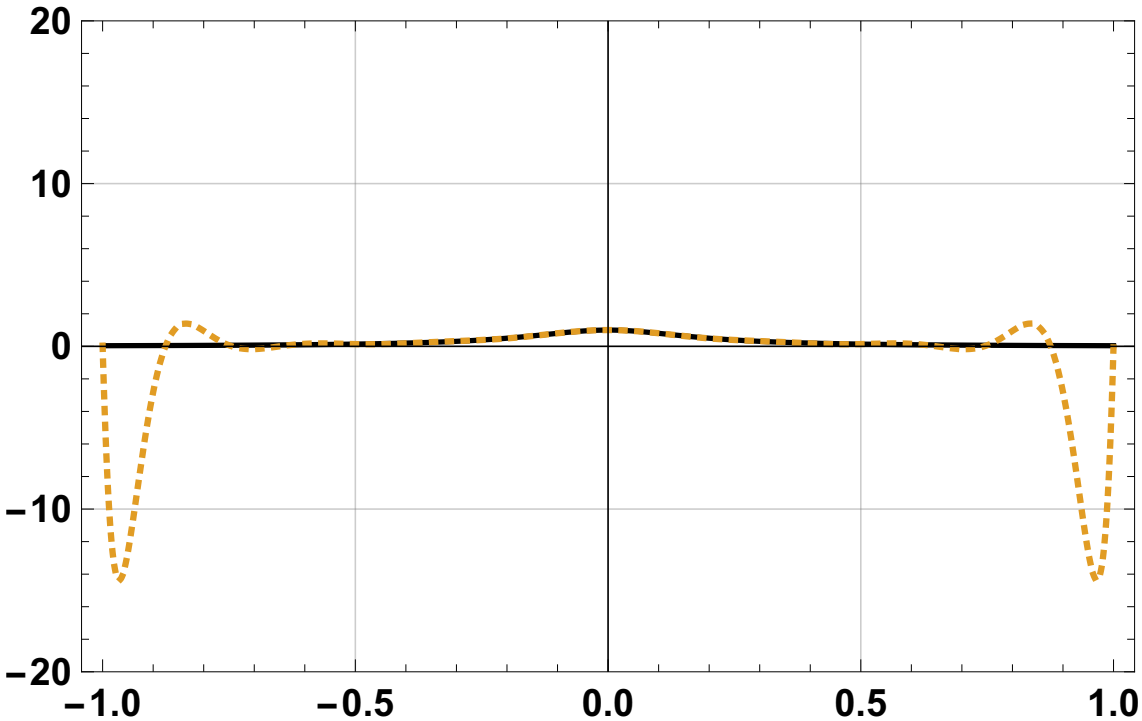
We will not prove this theorem, but we will do some numerical experiments to demonstrate the result. First, let's do some numerical experiment to test the hypothesis.



$$f(x) = \frac{1}{1 + 25x^2} \quad N = 8$$

Numerical Experiments

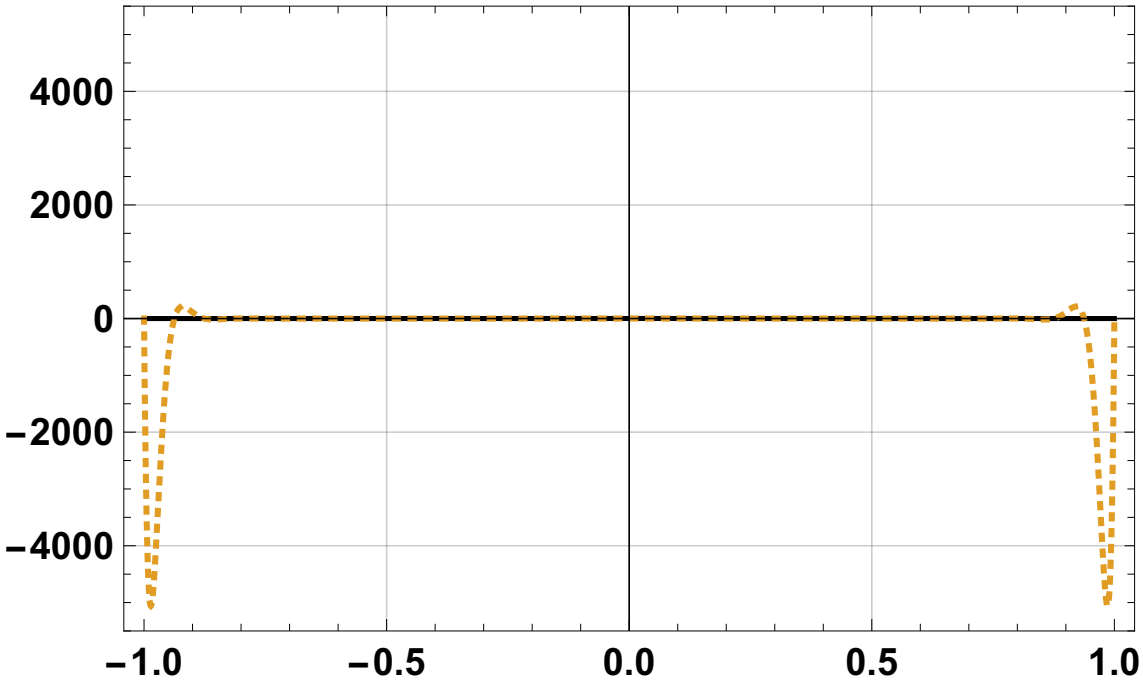
We will not prove this theorem, but we will do some numerical experiments to demonstrate the result. First, let's do some numerical experiment to test the hypothesis.



$$f(x) = \frac{1}{1 + 25x^2} \quad N = 16$$

Numerical Experiments

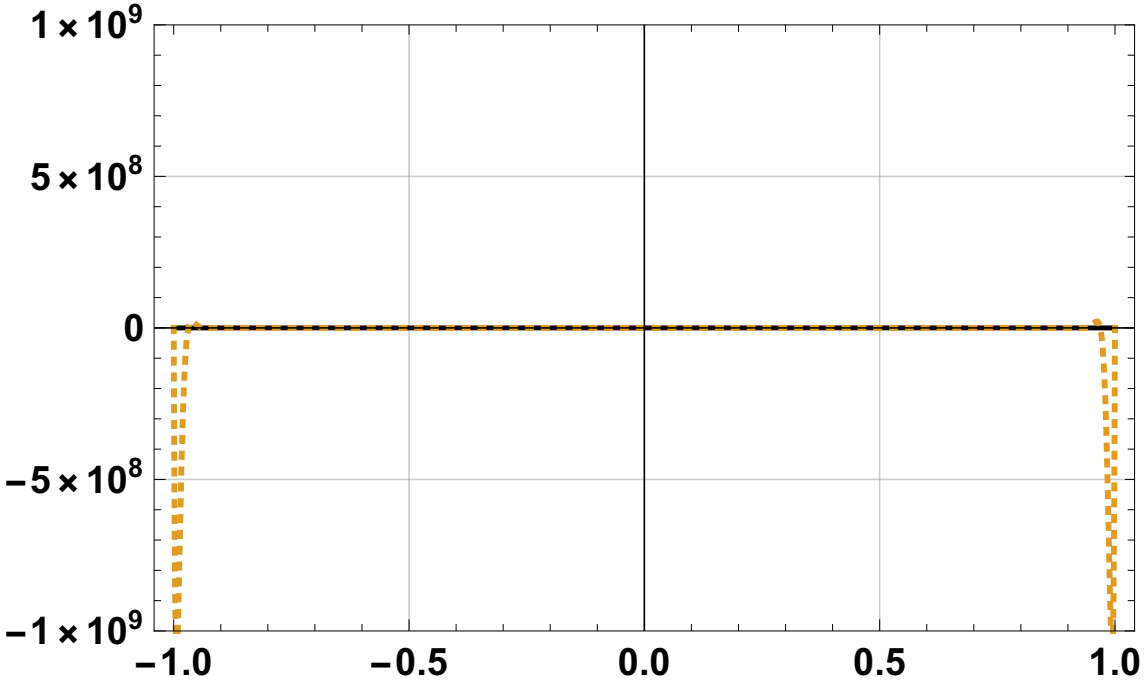
We will not prove this theorem, but we will do some numerical experiments to demonstrate the result. First, let's do some numerical experiment to test the hypothesis.



$$f(x) = \frac{1}{1 + 25x^2} \quad N = 32$$

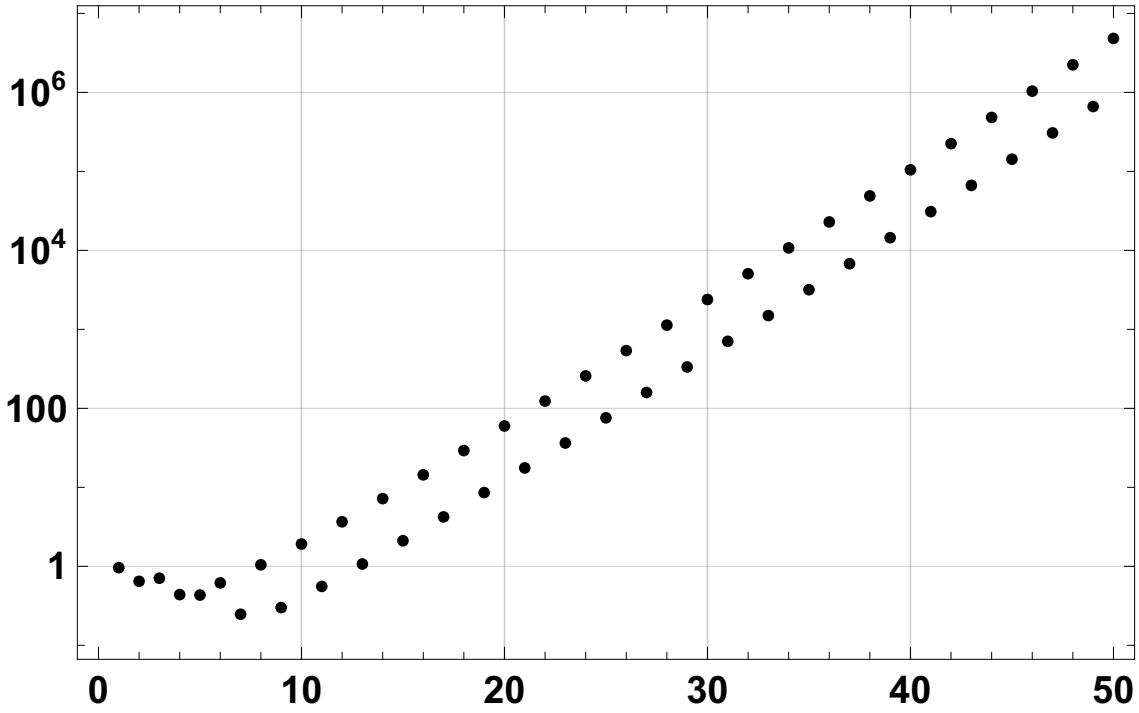
Numerical Experiments

We will not prove this theorem, but we will do some numerical experiments to demonstrate the result. First, let's do some numerical experiment to test the hypothesis.



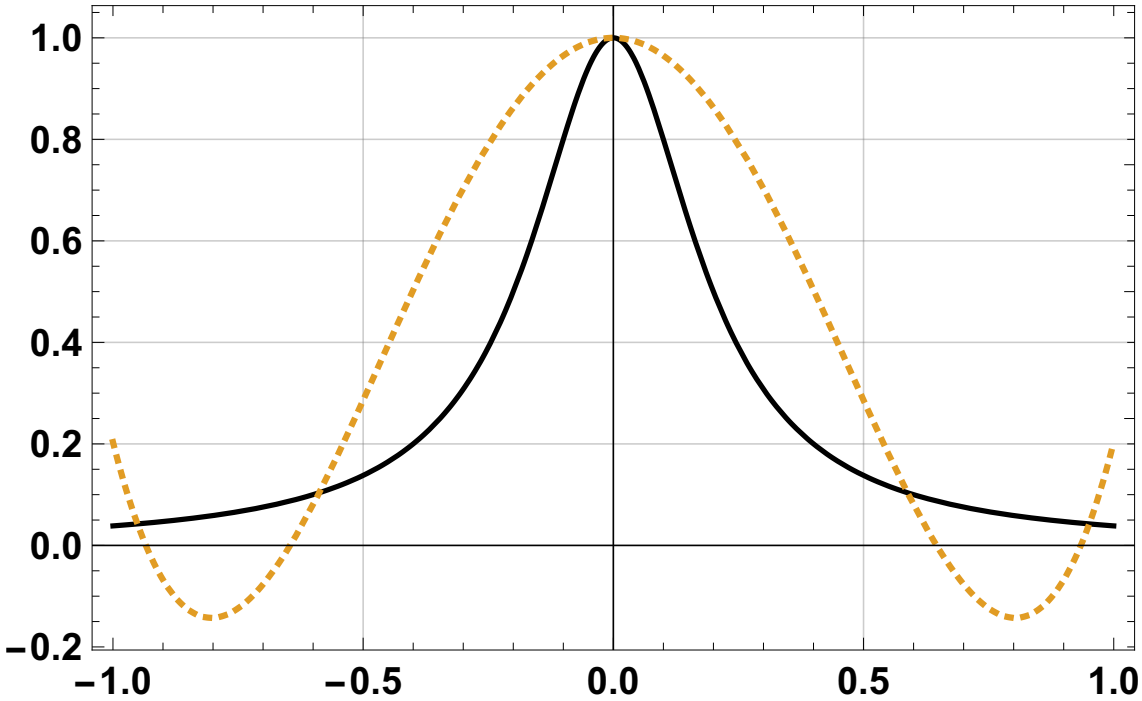
$$f(x) = \frac{1}{1 + 25x^2} \quad N = 64$$

Numerical Experiments



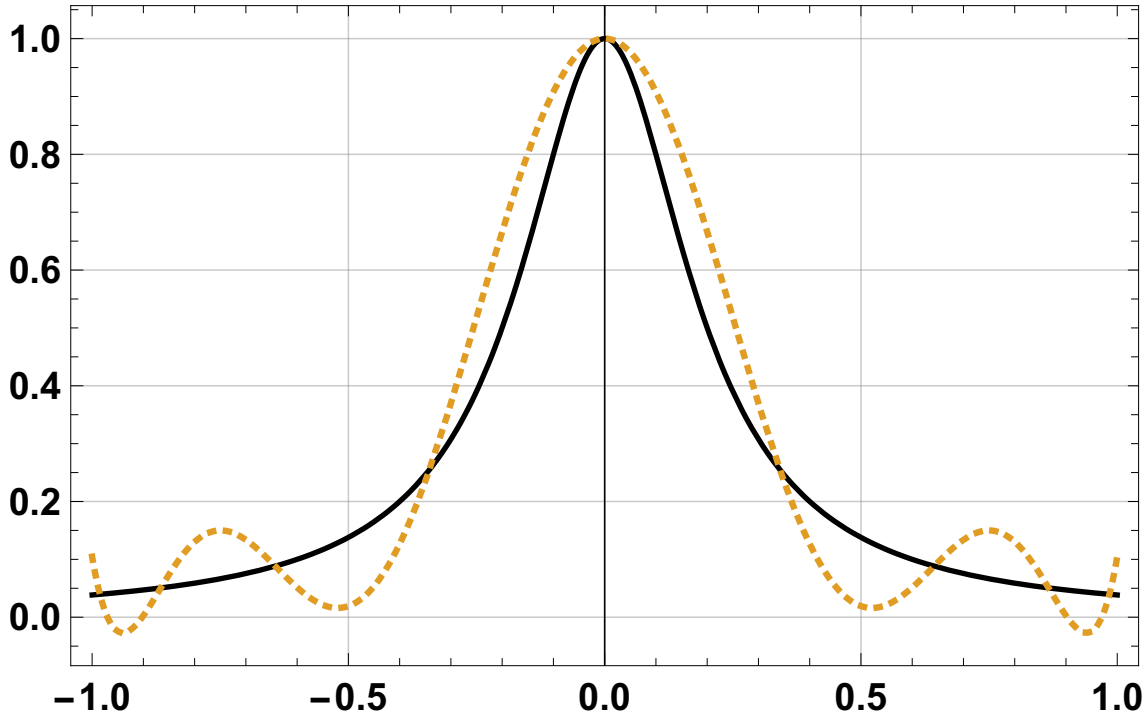
$\|Q_N - f(x)\|_\infty$ as a function of N.

The situation is quite different when we use Chebyshev interpolation nodes.



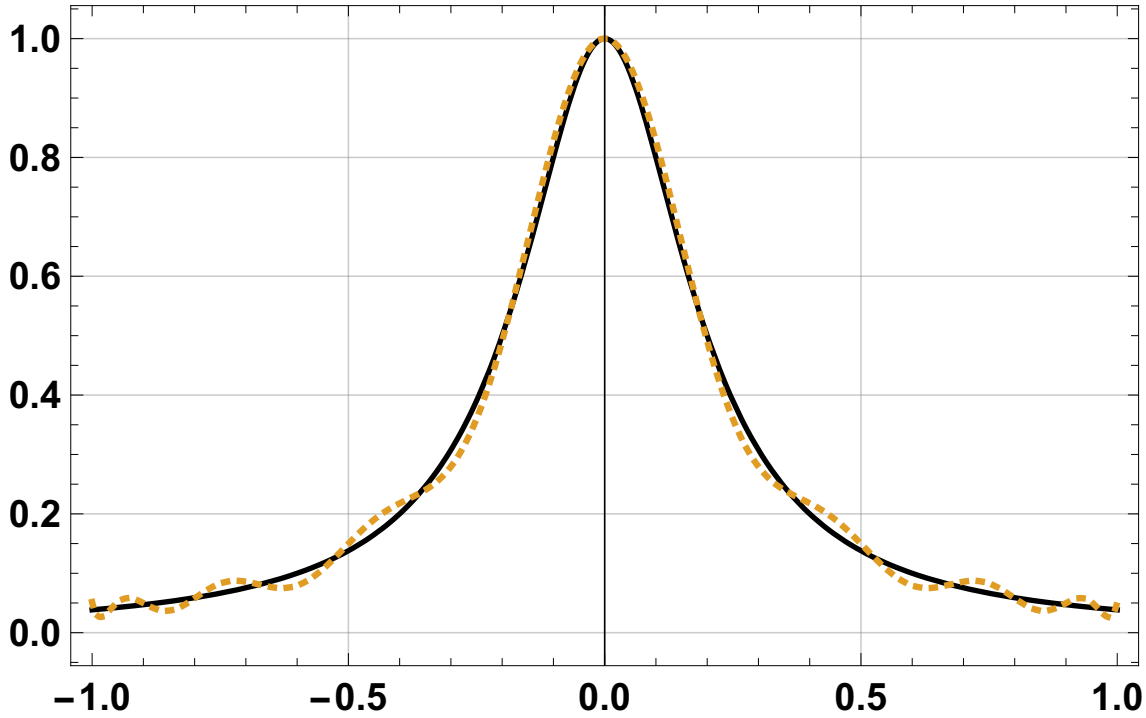
$$f(x) = \frac{1}{1 + 25x^2} \quad N = 4$$

The situation is quite different when we use Chebyshev interpolation nodes.



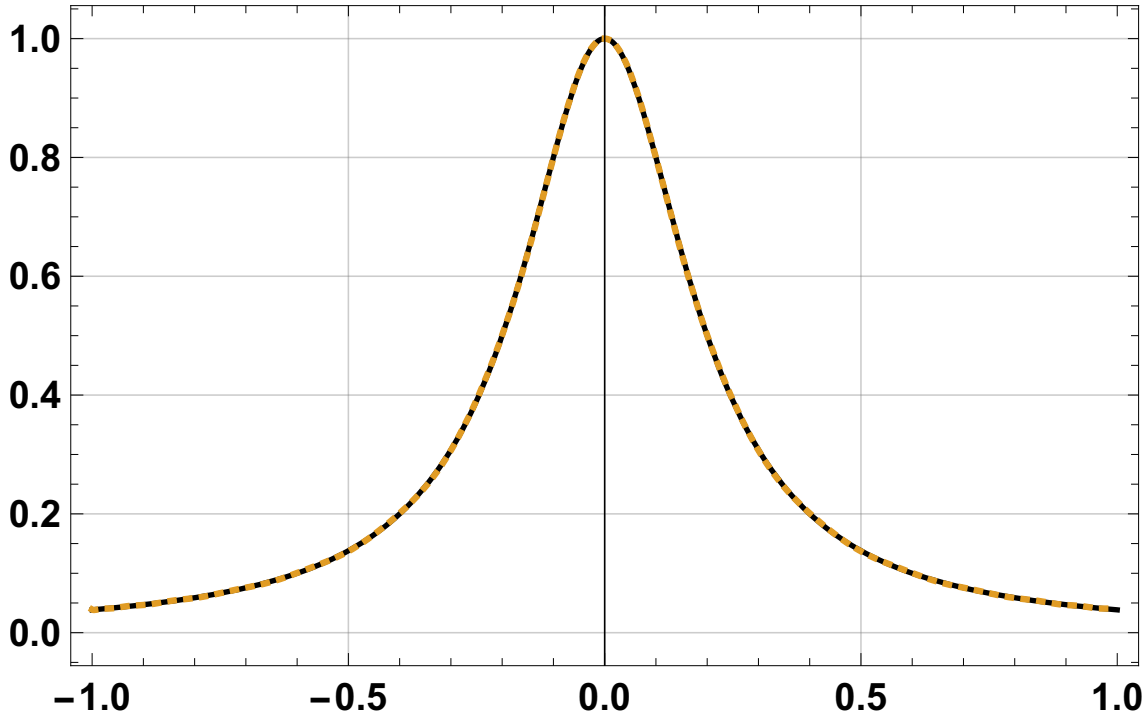
$$f(x) = \frac{1}{1+25x^2} \qquad N = 8$$

The situation is quite different when we use Chebyshev interpolation nodes.



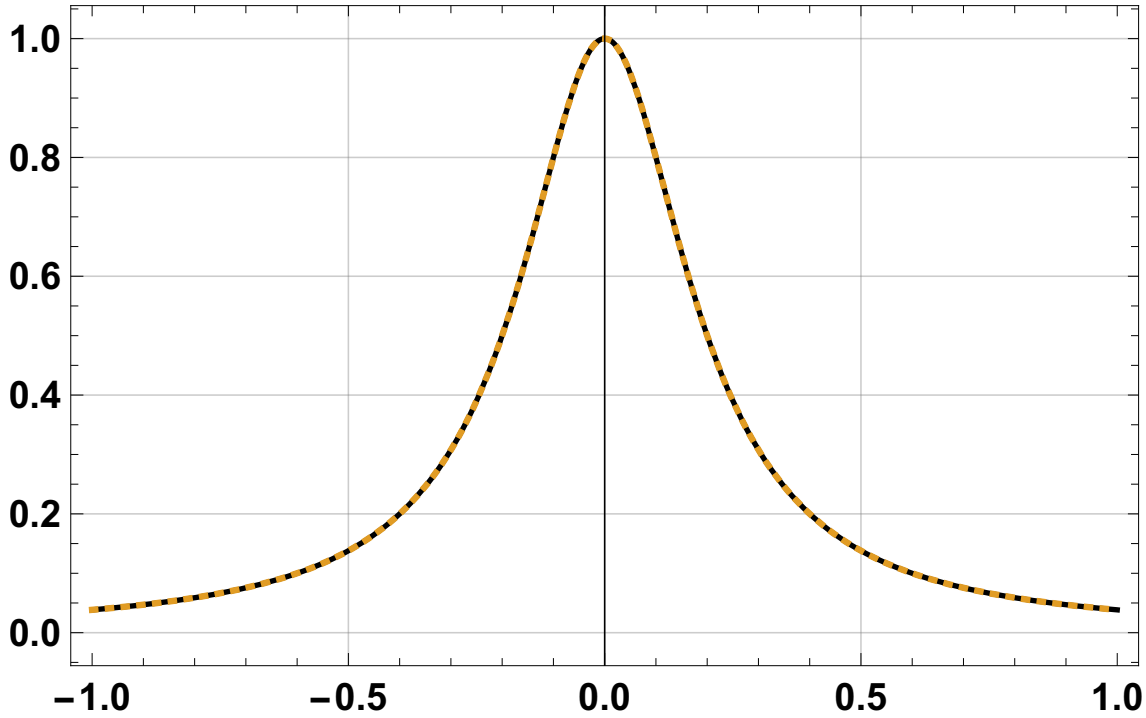
$$f(x) = \frac{1}{1 + 25x^2} \qquad N = 16$$

The situation is quite different when we use Chebyshev interpolation nodes.



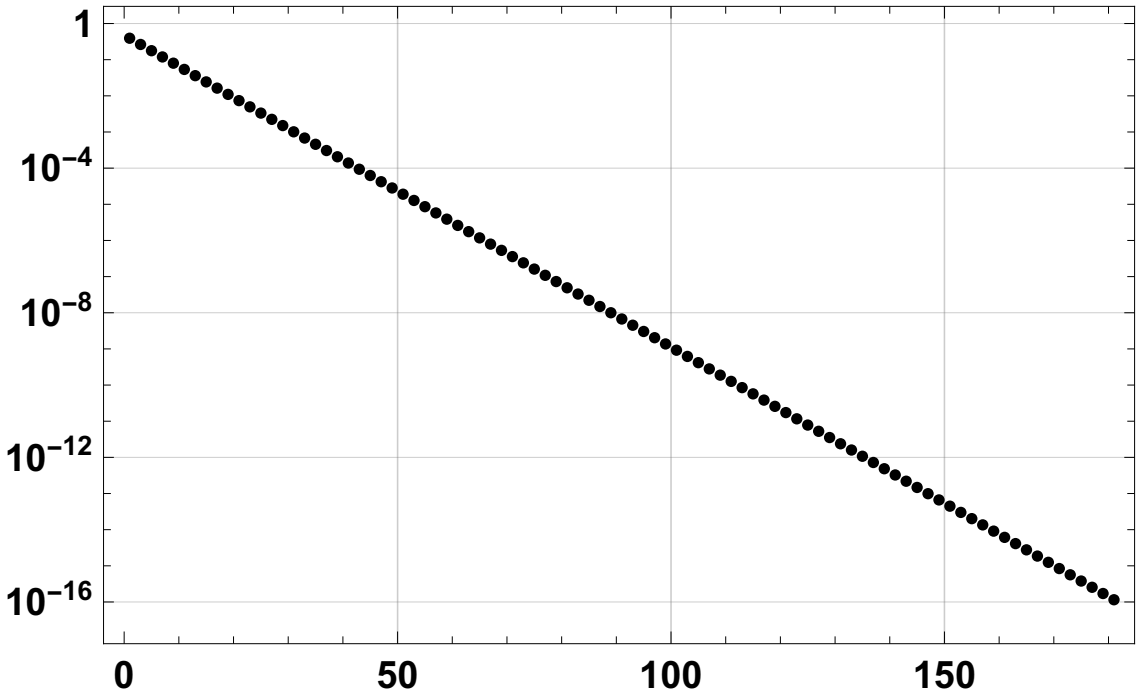
$$f(x) = \frac{1}{1 + 25x^2} \qquad N = 32$$

The situation is quite different when we use Chebyshev interpolation nodes.



$$f(x) = \frac{1}{1 + 25x^2} \qquad N = 64$$

Numerical Experiments



$\|P_N - f(x)\|_\infty$ as a function of N.

Why might use equispaced interpolation nodes despite all this?

We still often use equispaced interpolation.

Equispaced interpolation performs unusually poorly in the case of Runge's example $f(x) = \dfrac{1}{1+25x^3}$. In other cases, the equispaced inteprolation converges, albeit usually not as fast as in the case of Chebyshev interpolation.

To give an example, if we let $f(x) = \cos(x)$ and Q_N be the polynomial of degree N that interpolates f at the nodes

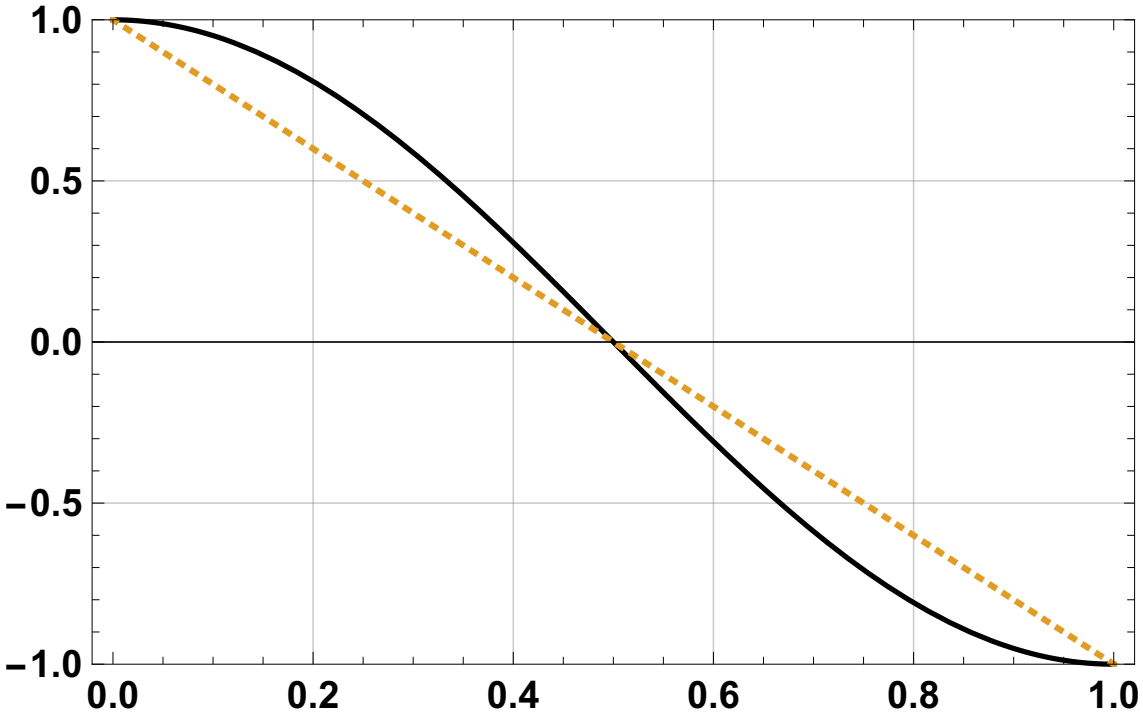
$$x_j = \frac{j}{N} \quad j = 0, 1 \ldots, N,$$

then from the interpolation error formula we proved earlier we see that

$$\begin{aligned} |f(x) - Q_N(x)| &= \left| \frac{f^{N+1}(\xi)}{(N+1)!} \prod_{j=0}^{N} (x - x_j) \right| \\ &\leq \frac{1}{(N+1)!}, \end{aligned}$$

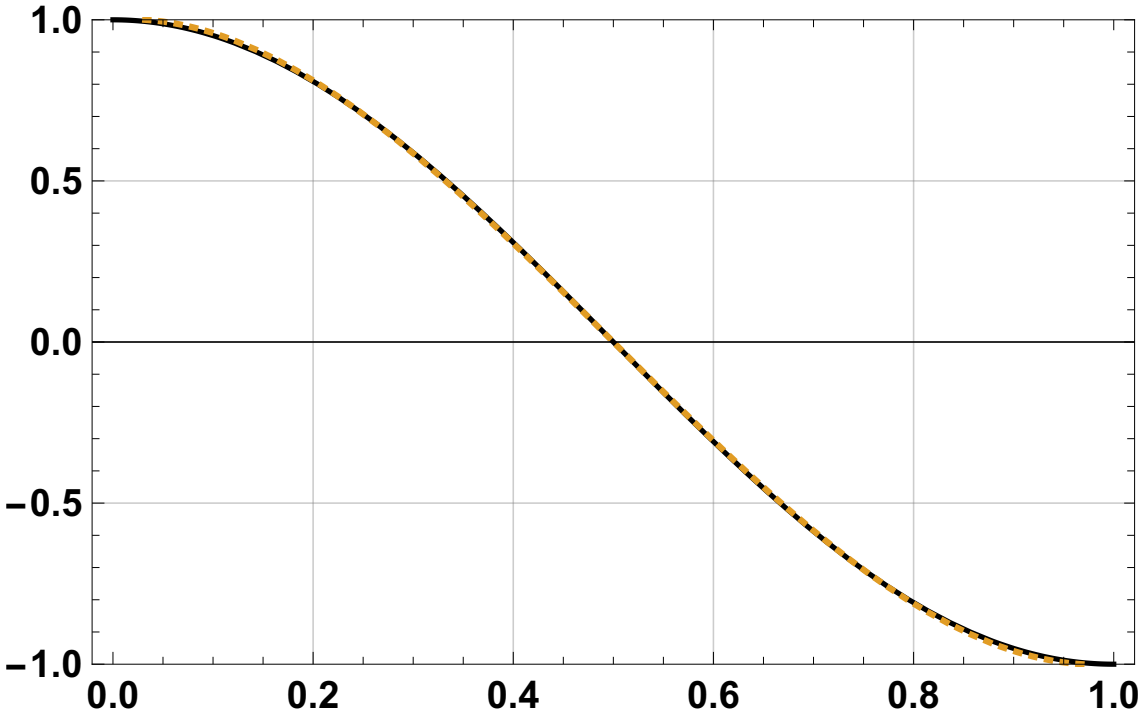
so Q_N does converge to f.

Why might use equispaced interpolation nodes despite all this?



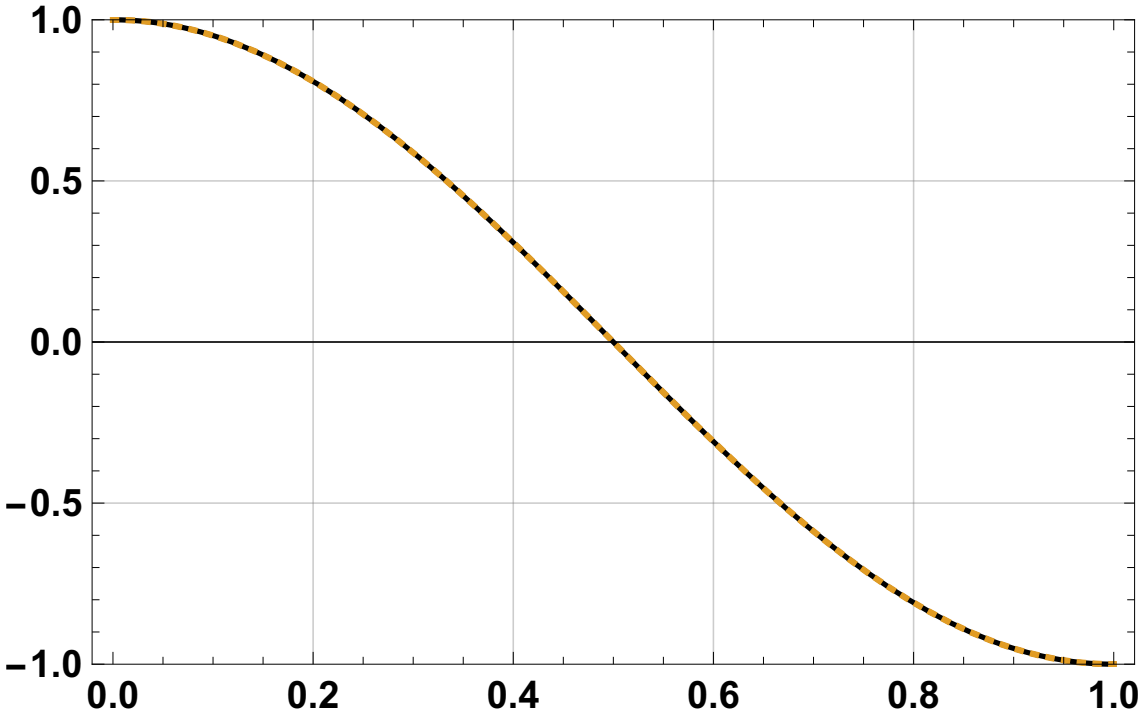
$f(x) = \cos(x) \quad N = 2$

Why might use equispaced interpolation nodes despite all this?



$f(x) = \cos(x) \quad N = 4$

Why might use equispaced interpolation nodes despite all this?



$f(x) = \cos(x) \quad N = 8$

Why might use equispaced interpolation nodes despite all this?

In many cases, we cannot control at what points we know the values of f. For instance, we might get data from an experiment and that experiment might be designed to give values only at equispaced nodes (a very common state of affairs).

We might deliberately place points at equispaced nodes. When using finite differences methods to approximate the derivatives of functions, the formulas are much simpler when equispaced nodes are used. This advantage can outweigh the disadvantages which arise from using equispaced nodes.

Moreover, interpolation from equispaced points is quite effective as long as we stay near the middle of interpolation interval. You will note that in the case of Runge's example the errors occurred near ± 1.

Finally, interpolation at equispaced nodes is usually fine at **low orders.** If we only use a small number of interpolation nodes, then the result errors are roughly the same as if we use Chebyshev nodes. When we do this, we usually use piecewise representations (splines) because we cannot hope to approximate complicated functions using low order polynomials.

MAT128A: Numerical Analysis
Lecture Seventeen: The Barycentric Form of the Lagrange Formula

November 2, 2018

Given a function $f : [a, b] \to \mathbb{R}$ and a positive integer N, the code you wrote as a part of the third programming project approximated the Chebyshev coefficients of f using the quadrature rule whose nodes are the roots of T_{N+1}. More explicitly, it computed the approximate coefficient defined by the formula

$$\widetilde{a}_n = \frac{2}{N+1} \sum_{j=0}^{N} f\left(\frac{b-a}{2} x_j + \frac{b+a}{2}\right) T_n(x_j),$$

where

$$x_j = \cos\left(\frac{j+\frac{1}{2}}{N+1}\right) \quad \text{for all} \quad j = 0, 1, \ldots, N.$$

Then, to approximate the value of f at a point x, you evaluated the expansion

$$\widetilde{p_N}(x) = {\sum_{j=0}^{N}}' \widetilde{a}_n T_n(x).$$

This process is equivalent to polynomial interpolation, but it involves an extra step.

Given the values of f at the nodes

$$x_0, x_1, \ldots, x_N$$

we first compute the coefficients

$$\widetilde{a_0}, \ldots, \widetilde{a_N}$$

and then we use those coefficients to approximate $f(x)$ using the formula

$$f(x) \approx {\sum_{j=0}^{N}}' \widetilde{a_n} T_n(x).$$

The first of these two steps requires $\mathcal{O}\left(N^2\right)$ operations, while the seconds requires $\mathcal{O}(N)$ operations.

But we can use the the Lagrange interpolation formula to evaluate $\widetilde{P_N}$ using the values of f at the nodes

$$x_0, x_1, \ldots, x_N$$

without calculating the coefficients

$$\widetilde{a_0}, \ldots, \widetilde{a_N}.$$

That is, we could simply use the formula

$$\widetilde{P_N}(x) = \sum_{i=0}^{N} f(x_i) L_i(x),$$

where

$$L_i(x) = \prod_{\substack{0 \le j \le N \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

Why might we not want to do this?

Why might we not want to do this?

Because it takes $\mathcal{O}\left(N^2\right)$ operations to evaluate $\widetilde{P_N}$ at a point x using the Lagrange formula.

Why might we not want to do this?

Because it takes $\mathcal{O}\left(N^2\right)$ operations to evaluate $\widetilde{P_N}$ at a point x using the Lagrange formula.

On the other hand, while computing the coefficients $\{\tilde{a_n}\}$ takes $\mathcal{O}\left(N^2\right)$ operations, once they are known evaluating $\widetilde{P_N}(x)$ costs $\mathcal{O}\left(N\right)$ operations.

Fortunately, we can rearrange the Lagrange formula so that the total operation count is $\mathcal{O}(N)$ rather than $\mathcal{O}(N^2)$!

Theorem

Suppose that N is a positive integer, and that $x_0, x_1, \ldots, x_N$ are distinct points in $\mathbb{R}$. Suppose also that

$$L(x) = \prod_{j=0}^{N} (x - x_j)$$

and that, for each $i = 0, 1, \ldots, N$, L_i is defined via

$$L_i(x) = \prod_{\substack{0 \leq j \leq N \\ j \neq i}} \frac{x - x_j}{x_i - x_j}.$$

Then

$$L_i(x) = \frac{L(x)}{L'(x_i)(x - x_i)}.$$

Proof: We can obtain a formula for $L'(x_i)$ by applying the "generalized product rule," which says that

$$\begin{aligned}\frac{d}{dx}\left(f_0(x)f_1(x)\cdots f_{N-1}(x)f_N(x)\right) &= f_0'(x)f_1(x)\cdots f_{N-1}(x)f_N(x)\\ &+ f_0(x)f_1'(x)\cdots f_{N-1}(x)f_N(x)\\ &+ \cdots\\ &+ f_0(x)f_1(x)\cdots f_{N-1}(x)f_N'(x).\end{aligned}$$

In particular, the derivative of

$$L(x) = (x - x_0)(x - x_1)\cdots(x - x_N)$$

is

$$L'(x) = \sum_{k=0}^{N} \prod_{j\neq k}(x - x_j)$$

When we plug $x = x_i$ into this formula, only the term with $k = i$ survives, so we have:

$$L'(x_i) = \sum_{k=0}^{N} \prod_{j\neq k}(x_i - x_j) = \prod_{j\neq i}(x_i - x_j).$$

It might be easier to see this if we consider the case $N = 2$ and $i = 1$. Then

$$L(x) = (x - x_0)(x - x_1)(x - x_2),$$

so that

$$L'(x) = (x - x_1)(x - x_2) + (x - x_0)(x - x_2) + (x - x_0)(x - x_1)$$

and

$$\begin{aligned} L'(x_1) &= (x_1 - x_1)(x_1 - x_2) + (x_1 - x_0)(x_1 - x_2) + (x_1 - x_0)(x - x_1) \\ &= (x_1 - x_0)(x_1 - x_2), \end{aligned}$$

which agrees with our formula:

$$L'(x_1) = \prod_{j \neq 1} (x_1 - x_j).$$

Now

$$\frac{L(x)}{L'(x_i)(x-x_i)} = \frac{\prod_{i=0}^{N}(x-x_i)}{(x-x_i)\prod_{j\neq i}(x_i-x_j)}$$
$$= \prod_{j\neq i}\frac{x-x_i}{x_i-x_j},$$

which is the claim of the theorem. ■

Why is this interesting?

Why is this interesting?

Because we can reorganize the Lagrange interpolation formula to lower the number of operations required to evaluate it.

Let

$$\lambda_i = \frac{1}{L'(x_i)}.$$

Then

$$L_i(x) = \frac{L(x)}{L'(x_i)(x - x_i)} = L(x)\frac{\lambda_i}{(x - x_i)}$$

so that

$$\begin{aligned}
\widetilde{P_N}(x) &= \sum_{i=0}^{N} f(x_i) L_i(x) \\
&= \sum_{i=0}^{N} f(x_i) L(x) \frac{\lambda_i}{(x - x_i)} \\
&= L(x) \sum_{i=0}^{N} f(x_i) \frac{\lambda_i}{x - x_i}.
\end{aligned}$$

So we have

$$\widetilde{P_N}(x) = L(x) \sum_{i=0}^{N} f(x_i) \frac{\lambda_i}{x - x_i}.$$

where

$$L(x) = (x - x_0)(x - x_1) \cdots (x - x_N).$$

The polynomial L can be evaluated in $\mathcal{O}(N)$ operations, as can the sum

$$\sum_{i=0}^{N} f(x_i) \frac{\lambda_i}{x - x_i}.$$

So, assuming that values of $\lambda_0, \ldots, \lambda_N$ are known, so can perform the interpolation in $\mathcal{O}(N)$ operations rather than $\mathcal{O}\left(N^2\right)$ operations.

In the case of Chebyshev root nodes, there is even a nice formula for the λ_i:

Theorem

Suppose that $f : [-1, 1] \to \mathbb{R}$ is a continuous function, and that N is a positive integer. Let $x_0, x_1, \ldots, x_N$ be the Chebyshev points defined by the formula

$$x_j = \cos\left(\frac{j + \frac{1}{2}}{N+1}\pi\right),$$

and let $\widetilde{P_N}$ be the unique polynomial of degree N which interpolates f at the nodes $x_0, x_1, \ldots, x_N$. Then

$$\widetilde{P_N}(x) = L(x) \sum_{i=0}^{N} f(x_i) \frac{\lambda_i}{x - x_i},$$

where

$$\lambda_i = (-1)^i \sin\left(\frac{i + \frac{1}{2}}{N+1}\pi\right).$$

In the case of Chebyshev extrema nodes, the λ_i have a particularly simple form:

Theorem

Suppose that $f : [-1, 1] \to \mathbb{R}$ is a continuous function, and that N is a positive integer. Let $x_0, x_1, \ldots, x_N$ be the Chebyshev points defined by the formula

$$x_j = \cos\left(\frac{j\pi}{N}\right),$$

and let $\widetilde{P_N}$ be the unique polynomial of degree N which interpolates f at the nodes $x_0, x_1, \ldots, x_N$. Then

$$\widetilde{P_N}(x) = L(x) \sum_{i=0}^{N}{}'' (-1)^i \frac{f(x_i)}{x - x_i}.$$

There is a little problem with the preceeding formulas. Although, they are extremely convenient and allow us to interpolate functions in $\mathcal{O}(N)$ operations, they are numerically unstable. But this can be fixed rather easily!

We observe that

$$\sum_{i=0}^{N} L_i(x) = 1$$

since this is the unique polynomial of degree less than or equal to N which is equal to 1 at each of the nodes $x_0, \ldots, x_N$. But

$$L_i(x) = L(x)\frac{\lambda_i}{x - x_i},$$

so

$$1 = \sum_{i=0}^{N} L_i(x) = L(x)\sum_{i=0}^{N} \frac{\lambda_i}{x - x_i},$$

In particular,

$$\begin{aligned}
\widetilde{P_N}(x) &= \left(L(x) \sum_{i=0}^{N} f(x_i) \frac{\lambda_i}{x - x_i} \right) \Big/ 1 \\
&= \left(L(x) \sum_{i=0}^{N} f(x_i) \frac{\lambda_i}{x - x_i} \right) \Big/ \left(L(x) \sum_{i=0}^{N} \frac{\lambda_i}{x - x_i} \right) \\
&= \left(\sum_{i=0}^{N} f(x_i) \frac{\lambda_i}{x - x_i} \right) \Big/ \left(\sum_{i=0}^{N} \frac{\lambda_i}{x - x_i} \right)
\end{aligned}$$

This is called the barycentric form of the Lagrange interpolation formula, and it is numerically stable!

The unique polynomial of degree N which interpolates f at the points $x_0, x_1, \ldots, x_N$ is

$$\sum_{i=0}^{N} \frac{\lambda_i}{x - x_i} f(x_i) \Bigg/ \sum_{i=0}^{N} \frac{\lambda_i}{x - x_i},$$

where

$$\lambda_i = \frac{1}{L'(x_i)}$$

with

$$L(x) = (x - x_0)(x - x_1) \cdots (x - x_N).$$

This is called the barycentric form of the Lagrange formula.

Theorem

Suppose that $f : [-1, 1] \to \mathbb{R}$ *is a continuous function, and that* N *is a positive integer. Let* $x_0, x_1, \ldots, x_N$ *be the Chebyshev points defined by the formula*

$$x_j = \cos\left(\frac{j\pi}{N}\right),$$

and let $\widetilde{P_N}$ *be the unique polynomial of degree* N *which interpolates* f *at the nodes* $x_0, x_1, \ldots, x_N$. *Then*

$$\widetilde{P_N}(x) = \sum_{j=0}^{N}{}'' \frac{(-1)^j}{x - x_j} f(x_j) \Bigg/ \sum_{j=0}^{N}{}'' \frac{(-1)^j}{x - x_j}.$$

MAT128A: Numerical Analysis
Lecture Eighteen: Introduction to Numerical Integration

November 16, 2018

Numerical Integration

Quadrature rules are formulas of the form

$$\int_a^b f(x)\ dx \approx \sum_{j=1}^{N} f(x_j) w_j$$

which are used to approximate integrals.

We have seen several useful quadrature rules already in this class. One example is the trapezoidal rule

$$\int_{-\pi}^{\pi} f(t)\ dt \approx \frac{2\pi}{2N+1} \sum_{j=0}^{2N} f\left(-\pi + \frac{2\pi}{2N+1} j\right)$$

which we used to evaluate Fourier coefficients.

We will now discuss methods for constructing such formulas more systematically.

Methodology for Constructing Quadrature Rules

$$\int_a^b f(x)\ dx \approx \sum_{j=1}^{N} f(x_j) w_j. \tag{1}$$

All of our quadrature rules will be constructed by first choosing a collection of functions

$$f_1, \ldots, f_M$$

and then finding nodes $x_1, \ldots, x_N$ and weights $w_1, \ldots, w_N$ such that (1) holds exactly whenever f is equal to one of the functions f_j.

The quadrature rule will accurate integrate any function f such that

$$f(x) \approx \sum_{j=1}^{M} a_j f_j(x).$$

Usually (but not always), it will be the case that $N = M$.

The Trapezoidal Rule

As a simple example, let's rederive the trapezoidal rule on a single interval using this procedure.

We let f_1, f_2 be the functions

$$f_1(x) = 1 \quad \text{and} \quad f_2(x) = x$$

We will choose the nodes x_1 and x_2 to be the endpoints $x_1 = a$ and $x_2 = b$.

Now we wish to find weights w_1 and w_2 such that

$$\int_a^b f(x)\, dx \approx f(x_1)w_1 + f(x_2)w_2$$

holds exactly when $f = f_1$ and $f = f_2$.

The Trapezoidal Rule

$$\int_a^b f(x)\ dx \approx f(x_1)w_1 + f(x_2)w_2 \tag{2}$$

We get the following system of two linear equations in the two unknowns w_1 and w_2 by requiring that (2) hold when $f(x) = 1$ and $f(x) = x$:

$$(b-a) = \int_a^b 1\ dx = w_1 + w_2$$
$$\frac{b^2}{2} - \frac{a^2}{2} = \int_a^b x\ dx = aw_1 + bw_2$$

You can verify that the unique solution of this system is

$$w_1 = \frac{b-a}{2} \quad w_2 = \frac{b-a}{2}.$$

The Trapezoidal Rule

This gives us the final version of the trapezoidal rule on a single interval:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\left(f(a)+f(b)\right).$$

Simpson's Rule

Simpson's rule for approximating

$$\int_a^b f(x)\,dx$$

is obtained by letting

$$f_1(x) = 1, \quad f_2(x) = x, \quad f_3(x) = x^2$$

and choosing the nodes

$$x_0 = a, \quad x_1 = \frac{a+b}{2}, \quad \text{and} \quad x_2 = b.$$

Simpson's Rule

$$\int_a^b f(x)\,dx \approx w_1 f(a) + w_2 f\left(\frac{a+b}{2}\right) + w_3 f(b) \tag{3}$$

This gives us the linear system

$$(b-a) = \int_a^b 1\,dx = w_1 + w_2 + w_3$$
$$\frac{b^2}{2} - \frac{a^2}{2} = \int_a^b x\,dx = aw_1 + \frac{a+b}{2}w_2 + bw_3$$
$$\frac{b^3}{3} - \frac{a^3}{3} = \int_a^b x^2\,dx = a^2 w_1 + \left(\frac{a+b}{2}\right)^2 w_2 + b^2 w_3$$

You can verify that the unique solution of this system is

$$w_1 = \frac{b-a}{6} \quad w_2 = 2\frac{b-a}{3}, \quad w_3 = \frac{b-a}{6}.$$

Simpson's Rule

By construction, Simpson's rule

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)$$

is exact whenever f is of the form

$$f(x) = a_0 + a_1 x + a_2 x^2.$$

In fact, it is actually exact for functions of the form

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3.$$

Simpson's 3/8 Rule and Boole's Rule

The next quadrature rule in this sequence is Simpson's 3/8 Rule

$$\int_a^b f(x)\,dx \approx \frac{3}{8}\frac{b-a}{2}\left(f(a)+3f\left(\frac{2a+b}{3}\right)+3f\left(\frac{a+2b}{3}\right)+f(b)\right),$$

which is also exact for polynomials of degrees up to 3.

Boole's Rule is

$$\int_a^b f(x)\,dx \approx \frac{2}{45}\frac{b-a}{2}\left(7f(a)+32f\left(\frac{a+3b}{4}\right)+12f\left(\frac{2a+2b}{4}\right)\right.$$
$$\left.+32f\left(\frac{a+3b}{4}\right)+7f(b)\right),$$

and it is exact for polynomials of degrees up to 5.

Newton-Cotes Quadrature Rule

In more generality, the N-point Newton-Cotes quadrature rule on the interval $[a, b]$ is the quadrature rule of the form

$$\int_a^b f(x)\, dx \approx \sum_{j=0}^{N-1} f(x_j) w_j \tag{4}$$

with

$$x_j = \frac{(N-j)a + jb}{N}$$

and the weights $w_0, \ldots, w_{N-1}$ chosen so that (4) is exact for polynomials of degrees less than or equal to $N-1$.

Newton-Cotes Quadrature Rule

The weights in the Newton-Cotes rule

$$\int_a^b f(x)\ dx \approx \sum_{j=0}^{N-1} f(x_j) w_j$$

can be determined by solving the system

$$\int_a^b 1\ dx = \sum_{j=0}^{N-1} w_j$$

$$\int_a^b x\ dx = \sum_{j=0}^{N-1} x_j w_j$$

$$\int_a^b x^2\ dx = \sum_{j=0}^{N-1} x_j^2 w_j$$

$$\vdots$$

$$\int_a^b x^{N-1}\ dx = \sum_{j=0}^{N-1} x_j^{N-1} w_j$$

of N linear equations in the N unknowns $w_0, w_1, \ldots, w_{N-1}$.

Alternate Derivation via the Lagrange Interpolation Formula

There is another way of deriving these formulas via the Lagrange Interpolation Formula.

Let p be the polynomial of degree $N-1$ which interpolates the function f at the nodes $x_0, x_1, \ldots, x_{N-1}$. We know that

$$p(x) = \sum_{j=0}^{N-1} f(x_j) L_j(x),$$

where

$$L_j(x) = \prod_{\substack{0 \le i \le N-1 \\ i \neq j}} \frac{x - x_i}{x_j - x_i}.$$

So

$$\int_a^b p(x)\ dx = \sum_{j=0}^{N-1} f(x_j) \int_a^b L_j(x)\ dx.$$

Alternate Derivation via the Lagrange Interpolation Formula

If we let

$$w_j = \int_a^b L_j(x)\ dx,$$

then we have

$$\int_a^b p(x)\ dx = \sum_{j=0}^{N-1} f(x_j) w_j. \tag{5}$$

If f is a polynomial of degree at most $N-1$, then it must coincude with the interpolating polynomial p on the interval $[a, b]$, and we have

$$\int_a^b f(x)\ dx = \int_a^b p(x)\ dx = \sum_{j=0}^{N-1} f(x_j) w_j.$$

It follows that (5) is the Newton-Cotes rule of length N.

Error Estimates for the Trapezoidal Rule

We will now derive an error estimate for the trapezoidal rule. We will need the following theorem, though:

Theorem (Integral Mean Value Theorem)

Suppose that $f : [a, b] \to \mathbb{R}$ *and* $g : [a, b] \to \mathbb{R}$ *are continuous function, and that* g *does not change sign on* $[a, b]$*. Then there exists* $c \in (a, b)$ *such that*

$$\int_a^b f(x)g(x)\,dx = f(c)\int_a^b g(x)\,dx.$$

Theorem (Error estimate for the trapezoidal rule)

Suppose that $f : [a, b] \to \mathbb{R}$ *is a twice continuously differentiable. Then there exists* $c \in (a, b)$ *such that*

$$\int_a^b f(x)\, dx - \frac{b-a}{2}\left(f(a) + f(b)\right) = f''(c)\, \frac{(b-a)^3}{12}$$

Proof:

Let p be the polynomial of degree 1 which interpolates f at the points a and b. Then the trapezoidal rule is exact for p — that is,

$$\int_a^b p(x)\, dx = \frac{b-a}{2}\left(p(a) + p(b)\right),$$

and since $p(a) = f(a)$ and $p(b) = f(b)$, we have:

$$\int_a^b p(x)\, dx = \frac{b-a}{2}\left(f(a) + f(b)\right).$$

For each x, there exists a point ξ_x in $[a, b]$ such that

$$f(x) - p(x) = \frac{f''(\xi_x)}{2}(x-a)(x-b).$$

It follows that

$$\int_a^b (f(x) - p(x))\ dx = \int_a^b \frac{f''(\xi_x)}{2}(x-a)(x-b)\ dx. \tag{6}$$

Now

$$\int_a^b p(x)\ dx = \frac{b-a}{2}\left(f(a) + f(b)\right),$$

so (6) is equivalent to

$$\int_a^b f(x)\ dx - \frac{b-a}{2}\left(f(a) + f(b)\right) = \int_a^b \frac{f''(\xi_x)}{2}(x-a)(x-b)\ dx. \tag{7}$$

Since $(x-a)(x-b)$ does not change sign on the interval, there is a point c in (a,b) such that

$$\begin{aligned}
\int_a^b f(x)\,dx - \frac{b-a}{2}\left(f(a)+f(b)\right) &= \int_a^b \frac{f''(\xi_x)}{2}(x-a)(x-b)\,dx \\
&= \frac{f''(c)}{2}\int_a^b \left(x^2-(a+b)x+ab\right)\,dx \\
&= \frac{f''(c)}{2}\frac{(b-a)^3}{6} \\
&= f''(c)\frac{(b-a)^3}{12}.
\end{aligned}$$

■

Theorem (Error estimate for Simpson's rule)

Suppose that $f : [a, b] \to \mathbb{R}$ *is four times continuously differentiable. Then there exists* $c \in (a, b)$ *such that*

$$\int_a^b f(x)\, dx - \frac{b-a}{3}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) = f^{(4)}(c)\,\frac{(b-a)^5}{90}$$

Proof:

Let p denote the polynomial which interpolates f at the points a, b and $z = (a+b)/2$. We have

$$f(x) - p(x) = \frac{f'''(\xi)}{6}(x-a)(x-b)(x-z),$$

but we cannot apply the integral mean value theorem to get our result because

$$(x-a)(x-b)(x-z)$$

changes sign on (a, b). Developing an alternate approach which works in this case will be a homework exercise! ■

MAT128A: Numerical Analysis
Lecture Nineteen: Curtis-Clenshaw Quadrature Rules and Gauss-Legendre Quadrature Rules

November 16, 2018

Integrating a Chebyshev expansion

If

$$f(x) \approx {\sum_{n=0}^{N}}' a_n T_n(x),$$

then we can be easily approximate the integral of f using the formula

$$\int_{-1}^{1} T_n(x)\, dx = \begin{cases} 0 & \text{n odd} \\ \dfrac{2}{1-n^2} & \text{n even.} \end{cases}$$

More explicitly, we have

$$\begin{aligned} \int_{-1}^{1} f(x)\, dx &= {\sum_{n=0}^{N}}' a_n \int_{-1}^{1} T_n(x)\, dx \\ &= {\sum_{m=0}^{\lfloor N/2 \rfloor}}' \frac{2a_{2m}}{1-4m^2} \end{aligned}$$

Of course, we cannot count on being able to compute the exact Chebyshev coefficients.

A Procedure for Approximating the Integral of a Function

But, given the values of f at the Chebyshev nodes

$$x_j = \cos\left(\frac{j\pi}{N}\right), \quad j = 0, 1, \ldots, N,$$

we can first form the approximate Chebyshev expansion

$$f(x) \approx \sum_{n=0}^{N}{}'' \widetilde{a}_n T_n(x)$$

with coefficients given by

$$\widetilde{a}_n = \frac{2}{N} \sum_{j=0}^{N}{}'' f(x_j) T_n(x_j),$$

and then approximate the integral of f over $[-1, 1]$ as follows:

$$\int_{-1}^{1} f(x)\, dx \approx \sum_{n=0}^{N}{}'' \widetilde{a}_n \int_{-1}^{1} T_n(x)\, dx = \sum_{m=0}^{\lfloor N/2 \rfloor}{}'' \frac{2\widetilde{a_{2m}}}{1 - 4m^2}.$$

A Procedure for Approximating the Integral of a Function

Of course, we only need to compute the even coefficients

$$\widetilde{a_{2m}} = \frac{2}{N} {\sum_{j=0}^{N}}'' f(x_j) T_{2m}(x_j), \quad m = 0, 1 \ldots, \lfloor N/2 \rfloor$$

to evaluate the formula

$$\int_{-1}^{1} f(x)\, dx \approx {\sum_{m=0}^{\lfloor N/2 \rfloor}}'' \frac{2\widetilde{a_{2m}}}{1 - 4m^2}.$$

Curtis-Clenshaw Quadrature Rules

Combining these two expressions gives us

$$\int_{-1}^{1} f(x)\, dx \approx \sum_{m=0}^{\lfloor N/2 \rfloor}{}'' \frac{2}{1-4m^2} \frac{2}{N} \sum_{j=0}^{N}{}'' f(x_j) T_{2m}(x_j)$$

$$= \sum_{j=0}^{N}{}'' f(x_j) \left(\frac{4}{N} \sum_{m=0}^{\lfloor N/2 \rfloor}{}'' \frac{T_{2m}(x_j)}{1-4m^2} \right)$$

Curtis-Clenshaw Quadrature Rules

If we define

$$w_j = \begin{cases} \frac{2}{N} \sum_{m=0}^{\lfloor N/2 \rfloor}{}'' \frac{T_{2m}(x_j)}{1-4m^2} & \text{if } j = 0, N \\ \frac{4}{N} \sum_{m=0}^{\lfloor N/2 \rfloor}{}'' \frac{T_{2m}(x_j)}{1-4m^2} & \text{otherwise,} \end{cases}$$

then we have the quadrature rule

$$\int_{-1}^{1} f(x)\, dx \approx \sum_{j=0}^{N} f(x_j) w_j$$

Recall that

$$x_j = \cos\left(\frac{j\pi}{N}\right), \quad j = 0, 1, \ldots, N.$$

By construction, this quadrature rule is exact for polynomials of degree less than or equal to N.

Curtis-Clenshaw Quadrature Rule

The quadrature rule

$$\int_{-1}^{1} f(x)\, dx \approx \sum_{j=0}^{N} f(x_j) w_j,$$

where, for each $j = 0, 1, \ldots, N$, x_j and w_j are defined by the formulas

$$x_j = \cos\left(\frac{j\pi}{N}\right)$$

and

$$w_j = \begin{cases} \frac{2}{N} \sum_{m=0}^{\lfloor N/2 \rfloor}{}'' \frac{T_{2m}(x_j)}{1-4m^2} & \text{if } j = 0, N \\ \frac{4}{N} \sum_{m=0}^{\lfloor N/2 \rfloor}{}'' \frac{T_{2m}(x_j)}{1-4m^2} & \text{otherwise,} \end{cases}$$

is exact for polynomials of degree less than or equal to N. It is known as the $(N+1)$-point Clenshaw-Curtis rule.

Exercise: Derive the analog of this rule for the root Chebyshev grid.

Accuracy of Curtis-Clenshaw Rules

Estimates on the accuracy of Curtis-Clenshaw rules can be developed using estimates on the rates of decay of the Chebyshev coefficients of various types of functions.

For instance, we expect Curtis-Clenshaw rules to converge like

$$\mathcal{O}\left(\frac{1}{n^k}\right)$$

when f is C^k and at an exponential rate when f is analytic.

Legendre Polynomials

Recall that Chebyshev polynomials are orthogonal with respect to the weight function $w(x) = (1 - x^2)^{-1/2}$. That is,

$$\int_{-1}^{1} T_n(x) T_m(x) \, \frac{dx}{\sqrt{1 - x^2}} = \begin{cases} 0 & n \neq m \\ \pi & n = m = 0 \\ \frac{\pi}{2} & n = m \neq 0. \end{cases}$$

The Legendre polynomails $\{P_n\}$ are a collection of polynomials which are orthogonal with respect to $w(x) = 1$:

$$\int_{-1}^{1} P_n(x) P_m(x) \, dx = \begin{cases} 0 & n \neq m \\ \frac{2}{2n+1} & n = m. \end{cases}$$

Legendre Polynomials

We will not investigate their properties too deeply, we will mostly rely on the following:

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ (n+1)P_{n+1}(x) &= (2n+1)xP_n(x) - nP_{n-1}(x). \end{aligned}$$

These formulas give us everything we need to evaluate P_n; the third formula is an example of a three-term recurrence relation.

Recall that the Chebyshev polynomials also satisfy similar relations:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2xT_n(x) - T_{n+1}(x). \end{aligned}$$

Gauss-Chebyshev quadrature rule

Earlier in the course, we saw a quadrature rule of the form

$$\int_{-1}^{1} f(x) \frac{dx}{\sqrt{1-x^2}} \approx \sum_{n=0}^{N} f(x_j) w_j$$

which exactly integrated polynomials of degree less than or equal to $2N+1$. More explicitly,

$$\int_{-1}^{1} f(x) \frac{dx}{\sqrt{1-x^2}} \approx \frac{\pi}{N+1} \sum_{j=0}^{N} f\left(\cos\left(\frac{j+\frac{1}{2}}{N+1}\pi\right)\right) T_m\left(\cos\left(\frac{j+\frac{1}{2}}{N+1}\pi\right)\right)$$

is exact whenever f is a polynomial of degree $2N+1$ or less. Notice that this $(N+1)$-point quadrature rule has the roots of T_{N+1} as its nodes.

Gauss-Legendre quadrature rule

The $(N+1)$-point Gauss-Legendre quadrature rule is obtained by letting

$$x_0, \ldots, x_N$$

be the $(N+1)$ roots of P_{N+1} and choosing weights $w_0, \ldots, w_N$ such that

$$\int_{-1}^{1} f(x)\, dx \approx \sum_{j=0}^{N} f(x_j) w_j$$

holds exactly when f is a polynomial of degree less than or equal to N.

Remarkably, this quadrature rule will turn out, like the Gauss-Chebyshev rule, to hold exactly for polynomials of degree less than or equal $2N+1$. We will demonstrate this now.

The key to our proof is:

Theorem (Polynomial Remainder Theorem)

Given any two polynomials p and q, there exist polynomials s and r such that the degree of r is strictly less than the degree of q and

$$p(x) = q(x)s(x) + r(x).$$

Suppose that p is a polynomial of degree $2N$. Then we can write

$$p(x) = q_{N-1}(x)P_{N+1}(x) + r_N(x),$$

where P_{N+1} is the Legendre polynomial of degree $(N+1)$, q_{N-1} a polynomial of degree at most $N-1$ and r_N a polynomial of degree at most N.

Since P_{N+1} is orthogonal to any polynomial of degree N or less, we have

$$\int_{-1}^{1} P_{N+1}(x)q_{N-1}(x)\ dx = 0.$$

It follows that

$$\int_{-1}^{1} p(x)\ dx = \int_{-1}^{1} (q_{N-1}(x)P_{N+1}(x) + r_N(x))\ dx = \int_{-1}^{1} r_N(x)\ dx.$$

Gauss-Legendre Quadrature Rules

Since the nodes x_j are the roots of P_{N+1}, we have

$$\sum_{j=0}^{N} p(x_j)w_j = \sum_{j=0}^{N} \left(q_{N-1}(x_j)P_{N+1}(x_j) + r_N(x_j)\right) w_j = \sum_{j=0}^{N} r_N(x_j)w_j.$$

But this quadrature rule is exact for polynomials of degree less than or equal to N — and r_N is of degree $\leq N$ — so

$$\sum_{j=0}^{N} p(x_j)w_j = \sum_{j=0}^{N} r_N(x_j)w_j = \int_{-1}^{1} r_N(x)\ dx.$$

Putting our two expressions together, we see that

$$\int_{-1}^{1} p(x)\ dx = \int_{-1}^{1} r_N(x)\ dx = \sum_{j=0}^{N} p(x_j)w_j,$$

so that the rule is exact for polynomials of degrees up to $2N$.

MAT128A: Numerical Analysis
Lecture Twenty: Numerical Differentiation I

November 16, 2018

Numerical quadrature

In the last few classes, we have been discussing the problem of constructing quadrature rules for approximating integrals of the form

$$\int_{-1}^{1} f(x)\ dx.$$

We came up with a reasonably good approach: let $x_0, x_1, \ldots, x_N$ be Chebyshev nodes or Legendre nodes and construct weights $w_0, \ldots, w_N$ such that

$$\int_{-1}^{1} f(x)\ dx \approx \sum_{j=0}^{N} f(x_j) w_j$$

is exact for polynomials of degrees between 0 and N. In the case of Chebyshev nodes, the resulting quadrature rule is known as a **Curtis-Clenshaw quadrature**. In the case of Legendre nodes, it is a **Gauss-Legendre quadrature**.

Numerical differentiation

In this lecture, we are going to bebgin discussing numerical differentiation.

That is, we will discuss methods for computing the derivative of a function f at a point x given its values at a collection of points near x, which may or may not include x.

Usually, we will be given a collection of points

$$x_0 < x_1 < x_2 < \cdots < x_n$$

and we will be interested in computing the values of $\{f'(x_j)\}$ given the values $\{f(x_j)\}$.

Numerical differentiation

Since

$$f'(x) = \lim_{h\to} \frac{f(x+h) - f(x)}{h},$$

an obvious approach is the approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

with h is a small positive real number.

Our first task will be to figure out how well this works in theory and in practice.

Numerical differentiation

If f is twice continuously differentiable, then by Taylor's theorem

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(c)$$

with c a point in the interval $(x, x+h)$. So

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{h}{2}f''(c)$$

and the error in our approximation is $\mathcal{O}(h)$.

Numerical differentiation

$$f(x) = \sin(x), \quad x = \frac{\pi}{4}$$

h	relative error
1×10^{-1}	0.0516241807514606 1
1×10^{-2}	0.0050166249167989 79
1×10^{-3}	0.0005001666248861964
1×10^{-4}	5.000166620515219e-05
1×10^{-5}	5.000010584208545e-06
1×10^{-6}	4.998429808212735e-07
1×10^{-7}	5.063952839184551e-08
1×10^{-8}	4.313707662086615e-09
1×10^{-9}	5.141648142259986e-08
1×10^{-10}	1.30749044836962e-06
1×10^{-11}	7.587860283104719e-06
1×10^{-12}	8.113064303733 03e-06
1×10^{-13}	0.0001488961815646445
1×10^{-14}	0.004859173557615 97

Numerical differentiation

So our approximation formula works, up to a point. The convergence is about what we expect, but we run into numerical trouble when h is small.

This isn't surprising, when h is small, the division by h in the formula

$$\frac{f(x+h)-f(x)}{h}$$

magnifies any roundoff errors in the computation of

$$f(x+h)-f(x).$$

Numerical differentiation

We can actually analyze the situation fairly easily.

The roundoff error in $f(x+h)-f(x)$ is on the order of machine precision, let's call that ϵ.

Let's let F denote the value of

$$\frac{f(x+h)-f(x)}{h}$$

obtained when we perform the operations in exact arithmetic so that

$$F - f'(x) = \mathcal{O}(h).$$

We let $\widetilde{F}$ denote the value obtained when we perform these operations numerically. Then:

$$F - \widetilde{F} = \mathcal{O}\left(\frac{\epsilon}{h}\right),$$

where ϵ is machine precision. So we have

$$f'(x) - \widetilde{F} = \left(F - \widetilde{F}\right) + \left(f'(x) - F\right) = \mathcal{O}(h) + \mathcal{O}\left(\frac{\epsilon}{h}\right)$$

The formula

$$f'(x) - \widetilde{F} = \left(F - \widetilde{F}\right) + \left(f'(x) - F\right) = \mathcal{O}(h) + \mathcal{O}\left(\frac{\epsilon}{h}\right)$$

explains what happens in the preceeding table: the errors improve as long as the term

$$\mathcal{O}(h)$$

is dominant. But once h is small enough.

$$\mathcal{O}\left(\frac{\epsilon}{h}\right)$$

becomes dominant, and the errors start increasing.

There is no way around this. There is no stable way to compute derivatives numerically.

Numerical differentiation = unstable

Numerical integration = stable

Numerical differentiation

What is particularly distressing about these results is that our approach is **nonconvergent**.

When we compute

$$\int_{-1}^{1} f(x)\ dx$$

using Curtis-Clenshaw quadrature rules, we are guaranteed to eventually obtain the correct value if we use a large enough value of n.

By contrast, the approximation formula

$$\frac{f(x+h)-f(x)}{h} = f'(x) + \frac{h}{2}f''(c)$$

becomes more accurate as we descrease h up to a point, and then the error starts increasing. This makes it difficult to tell what value of h to use.

Numerical differentiation

Even if we accept the instability in the formula

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

it is a particularly effective mechanism for approximating derivatives. The error obtained using it is $\mathcal{O}(h)$.

If we are given the values of f at equispaced nodes on an interval, we will need many nodes to obtain f' to high accuracy using this formula.

A Higher Order Formula

Here is another mechanism we could use to approximate f':

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}.$$

Since

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{3!}f'''(c)$$

and

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{3!}f'''(c'),$$

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \mathcal{O}\left(h^2\right).$$

This is a second order accurate formula!

A Higher Order Formula

$$f(x) = \sin(x), \quad x = \frac{\pi}{4}$$

h	relative error
1×10^{-1}	0.0016658335317180 97
1×10^{-2}	1.666658333395761e-05
1×10^{-3}	1.666667886286781e-07
1×10^{-4}	1.666529087425379e-09
1×10^{-5}	2.061968991527391e-11
1×10^{-6}	5.129894792848848e-11
1×10^{-7}	3.596646968930109e-10
1×10^{-8}	4.777380863375732e-09
1×10^{-9}	5.496710257161753e-09
1×10^{-10}	1.082376214625366e-07
1×10^{-11}	2.163055845570034e-06
1×10^{-12}	2.2711238086645e-05
1×10^{-13}	0.0003309339717027695
1×10^{-14}	0.006860929812673469
1×10^{-15}	0.027409112053748 44

A Theme

You should probably have noticed a theme by now: we have been using Taylor's formula to establish error estimates for our formulas.

We can try to use Taylor's theorem to construct our formulas.

Suppose we want to build a second order accurate formula for $f'(x)$ given the values of f at x, $x+h$ and $x+2h$. How would we do that?

Since

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \mathcal{O}(h^3)$$

$$f(x+2h) = f(x) + 2hf'(x) + 4\frac{h^2}{2}f''(x) + \mathcal{O}(h^3)$$

we have

$$\begin{aligned} &af(x) + bf(x+h) + cf(x+2h) \\ &= (a+b+c)f(x) + (b+2c)hf'(x) + (b+4c)\frac{h^2}{2}f''(x) + \mathcal{O}(h^3). \end{aligned}$$

In order to get

$$f'(x) = (a+b+c)f(x) + (b+2c)hf'(x) + (b+4c)\frac{h^2}{2}f''(x) + \mathcal{O}(h^3)$$

we want

$$a+b+c = 0$$
$$b+2c = \frac{1}{h}$$
$$b+4c = 0$$

which implies

$$a = -\frac{3}{2h}$$
$$b = \frac{2}{h}$$
$$c = -\frac{1}{2h}$$

Here is another formula derived in this fashion:

$$f'(x) = \frac{1}{12h}f(x-2h) - \frac{2}{3h}f(x-h) + \frac{2}{3h}f(x+h) - \frac{1}{12h}f(x+2h) + \mathcal{O}\left(h^4\right)$$

$$f(x) = \sin(x), \quad x = \frac{\pi}{4}$$

h	relative error
1×10^{-1}	3.3293673915219 36e-06
1×10^{-2}	3.3332459852364 41e-10
1×10^{-3}	2.5130426880834 69e-13
1×10^{-4}	1.8534460381449 62e-13
1×10^{-5}	7.6340606662041 73e-12
1×10^{-6}	7.6984175729832 2e-11
1×10^{-7}	9.2459669317417 46e-10
1×10^{-8}	1.2482949203778 84e-08
1×10^{-9}	2.0188517544181 96e-08

Higher order derivatives

It is also useful to be able to approximate higher order derivatives, such as $f''(x)$ and $f'''(x)$.

Let's derive a formula for $f''(x)$ using the values of f at $x - h$, x and $x + h$.

Once again, the starting place is Taylor's formula.

Higher order derivatives

Since

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) + \mathcal{O}(h^3)$$
$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \mathcal{O}(h^3),$$

we have:

$$af(x-h) + bf(x) + cf(x+h)$$
$$= (a+b+c)f(x) + (c-a)hf'(x) + (a+c)\frac{h^2}{2}f''(x) + \mathcal{O}(h^3)$$

Higher order derivatives

We want:

$$f''(x) = (a+b+c)f(x) + (c-a)hf'(x) + (a+c)\frac{h^2}{2}f''(x),$$

which implies:

$$\begin{aligned} a+b+c &= 0 \\ c-a &= 0 \\ a+c &= \frac{2}{h^2}. \end{aligned}$$

The solution is

$$\begin{aligned} a &= \frac{1}{h^2} \\ b &= \frac{-2}{h^2} \\ c &= \frac{1}{h^2} \end{aligned}$$

Higher order derivatives

It turns out that

$$f''(x) = \frac{f(x-h) + f(x+h) - 2f(x)}{h^2} + \mathcal{O}\left(h^2\right)$$

MAT128A: Numerical Analysis
Lecture Twenty-one: Spectral Integration and Differentiation

December 3, 2018

Spectral differentiation

In many applications, we wish to compute the values of $f'(x)$ or some other derivative of f at the nodes

$$x_0 < x_1 < x_2 < \cdots < x_N$$

given the values of f at those same nodes.

We can certainly do this using finite differences methods, and that is the method of choice when the nodes are equispaced.

If we have control over the placement of the nodes, we can also consider spectral differentiation methods.

Spectral differentiation

The idea behind spectral differentiation is quite simple. If

$$f(x) = \sum_{n=0}^{N}{}' a_n T_n(x),$$

then we can compute f' by knowing the derivatives of T_n. In particular:

$$f'(x) = \sum_{n=0}^{N}{}' a_n T_n'(x),$$

Spectral differentiation

In Homework 4, you showed that

$$T_n'(x) = \frac{nT_{n-1}(x) - nxT_n(x)}{1-x^2}$$

for $n > 0$. Obviously, $T_0'(x) = 0$.

So we have

$$\begin{aligned} f'(x) &= {\sum_{n=0}^{N}}' a_n T_n'(x) \\ &= \sum_{n=1}^{N} a_n \frac{nT_{n-1}(x) - nxT_n(x)}{1-x^2}. \end{aligned}$$

Of course, this is assuming we have access to the exact Chebyshev coefficients. But we can use this idea to build a numerical method for evaluating f' given the values of f.

Spectral differentiation

Given the values $\{f(x_j)\}$ with $\{x_j\}$ the Chebyshev root grid, we can approximate f via the expansion

$$\sum_{n=0}^{N}{}' \widetilde{a}_n T_n(x),$$

where

$$\widetilde{a}_n = \frac{2}{N} \sum_{j=0}^{N} f(x_j) T_n(x_j).$$

Then

$$\begin{aligned} f'(x) &\approx \sum_{n=0}^{N}{}' \widetilde{a}_n T_n'(x) \\ &= \sum_{n=1}^{N} \widetilde{a}_n \frac{n T_{n-1}(x) - n x T_n(x)}{1 - x^2}. \end{aligned}$$

We can use this formula to calculate the values $\{f'(x_j)\}$.

Spectral differentiation

We can also reorganize this calculation as

$$
\begin{aligned}
f'(x_i) &\approx \sum_{n=1}^{N} \widetilde{a}_n \frac{n T_{n-1}(x_i) - n x_i T_n(x_i)}{1 - x_i^2} \\
&= \sum_{n=1}^{N} \left(\frac{2}{N} \sum_{j=0}^{N} f(x_j) T_n(x_j) \right) \frac{n T_{n-1}(x_i) - n x_i T_n(x_i)}{1 - x_i^2} \\
&= \frac{2}{N} \sum_{j=0}^{N} f(x_j) \left(\sum_{n=1}^{N} T_n(x_j) \frac{n T_{n-1}(x_i) - n x_i T_n(x_i)}{1 - x_i^2} \right)
\end{aligned}
$$

Spectral differentiation

If we define

$$S_{ij} = \left(\sum_{n=1}^{N} T_n(x_j) \frac{nT_{n-1}(x_i) - nx_i T_n(x_i)}{1 - x_i^2} \right),$$

then

$$f(x_i) = \sum_{j=0}^{N} f(X_j) S_{ij}.$$

In other words,

$$\begin{pmatrix} f'(x_0) \\ f'(x_1) \\ \vdots \\ f'(x_N) \end{pmatrix} = S \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_N) \end{pmatrix},$$

where S is the $(N+1) \times (N+1)$ matrix with entries S_{ij}. We call S the spectral differentiaion matrix of order $N+1$.

Spectral differentiation

Using the facts that $T_n(x) = \cos(n \arccos(x))$ and

$$x_i = \cos\left(\frac{i + \frac{1}{2}}{N+1}\pi\right),$$

we can simplify the expression for S_{ij}:

$$\begin{aligned} S_{ij} &= \left(\sum_{n=1}^{N} T_n(x_j) \frac{nT_{n-1}(x_i) - nx_i T_n(x_i)}{1 - x_i^2}\right) \\ &= \left(\sum_{n=1}^{N} \cos(nt_j) \frac{n\cos((n-1)t_i) - n\cos(t_i)\cos(nt_i)}{\sin^2(t_i)}\right) \end{aligned}$$

where

$$t_i = \frac{i + \frac{1}{2}}{N+1}\pi.$$

Spectral differentiation

I usually don't think about it this way. The way I would typically form the matrix S would be to first form the $(N+1) \times (N+1)$ matrix U which takes the values of f at the nodes $\{x_j\}$ to its coefficients $\widetilde{a_n}$ in the approximate Chebyshev expansion.

That is, U is the $(N+1) \times (N+1)$ matrix

$$U \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_N) \end{pmatrix} = \begin{pmatrix} \widetilde{a_0} \\ \widetilde{a_1} \\ \vdots \\ \widetilde{a_N} \end{pmatrix}.$$

Spectral differentiation

Since

$$\widetilde{a_n} = \frac{2}{N} \sum_{j=0}^{N} f(x_j) T_n(x_j),$$

we can determine the entries of U quite easily:

$$U_{ij} = \frac{2}{N} T_i(x_j) = \frac{2}{N} \cos\left(i \frac{j + \frac{1}{2}}{N+1} \pi \right)$$

Spectral differentiation

Next, we form the matrix D which takes the coefficients $\{\widetilde{a_n}\}$ to the values of the derivative of the Chebyshev expansion

$$\sum_{n=0}^{N}{}' \widetilde{a_n} T_n(x)$$

at the points $\{x_j\}$.

Again, the entries of that matrix are not too hard to form since

$$\begin{aligned} f'(x) &\approx \sum_{n=0}^{N}{}' \widetilde{a_n} T_n'(x) \\ &= \sum_{n=1}^{N} \widetilde{a_n} \frac{nT_{n-1}(x) - nxT_n(x)}{1-x^2}. \end{aligned}$$

Spectral differentiation

Indeed, from

$$f'(x) \approx {\sum_{n=0}^{N}}' \widetilde{a}_n T_n'(x)$$
$$= \sum_{n=1}^{N} \widetilde{a}_n \frac{nT_{n-1}(x) - nxT_n(x)}{1-x^2}.$$

we see that the entries of D are given by

$$D_{ij} = \begin{cases} 0 & j = 0 \\ \dfrac{jT_{j-1}(x_i) - jx_i T_j(x_i)}{1-x_i^2} & j \neq 0 \end{cases}$$

I leave it to you to write this expression in terms of cosine and sine.

Spectral differentiation

The spectral differentiation matrix S is the product of D with U. That is

$$S = DU.$$

Thinking this way is extremely useful for rapidly constructing any number of objects related to Chebyshev polynomials.

For instance, the weights in the Curtis-Clenshaw quadrature are the entries of the vector W given by

$$W = \begin{pmatrix} 2 & 0 & -\frac{2}{3} & 0 & -\frac{2}{15} & \cdots \end{pmatrix} U$$