Lecture 8: Approximation

Yuya Takahashi

May 14, 2019

1 Introduction

- Three types of approximation problems:

    1. Local Approximation: The data are the value of a function f and its derivatives at some point x_0. The goal is to construct a function that matches those properties at x_0 and is a good approximation of f near x_0. Think of Taylor series.

    2. L^p- approximations: Take a function f and find a function g that is close to f (over an interval) in the sense of the L^p-norm. Ideally we need the entire function f.

    3. Interpolation: Find a function g that approximates f over a collection of points. In particular we make f and g be the same at those points and hope that the rest of g is a good approximation for f. While the data for L^p-approximations are the whole function f, for interpolation it is only a collection of n points that helps us fix n parameters.

- Regression is between interpolation and L^p-approximations. It uses $m > n$ points to fix n parameters. They use more information than interpolation but less than L^p. The real attractiveness is that this makes it a pretty stable method.

- Functional approximations are widely used in application

  - Value functions are approximated
  - Not only when we discretize a problem with continuous state variables, we use approximations when the original problem is finite (example of dynamic discrete choice with large state spaces below)

- First several sections mostly focus on functions of a single variable.

2 Local Approximation Methods

- Use information (possibly including derivatives) about a function $f : \mathbb{R} \to \mathbb{R}$ only at a point $x_0 \in \mathbb{R}$.

- The goal is to produce a good approximation to f around x_0.

Tylor Series

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2} f''(x_0) + \cdots + \frac{(n - x_0)^n}{n!} f^{(n)}(x_0) + \mathcal{R}_{n+1}(x)$$

where $\mathcal{R}_{n+1}(x)$ is $\mathcal{O}(|x - x_0|^{n+1})$ (meaning that the remainder is asymptotically smaller than the leading terms).

- These are polynomials that agree with f up to a high order around x_0. The accuracy may decline pretty fast as we move away from x_0.

Theorem Let f be analytical at x. If f or any derivative of f has a singularity at z then the radius of convergence of the Taylor series based at x_0 is bounded above by $||x_0 - z||$.

- It tells us that the Taylor series at x_0 cannot approximate $f(x)$ reliably at any point farther from x_0 than any singular point.

- Notice it is **distance** from the singularity that matters and not direction.

- $f(x) = x^{-\frac{1}{2}}$ is singular at $x = 0$ so the Taylor series based on $x_0 = 1$ will not provide a reliable approximation either below $x = 0$ or above $x = 2$ even though there is no singularity there.

2.1 Log-Linearization

- Suppose we know a solution $f(x_0, \varepsilon_0) = 0$ and we want to compute the elasticity $\frac{dx/x}{d\varepsilon/\varepsilon}$ at $x = x_0$ and $\varepsilon = \varepsilon_0 \neq 0$.

- By implicit differentiation

$$d(\ln x) = \frac{dx}{x} = -\frac{\varepsilon f_\varepsilon}{x f_x}\frac{d\varepsilon}{\varepsilon} = -\frac{\varepsilon f_\varepsilon}{x f_x} d(\ln \varepsilon).$$

- This in turn implies the log linear approximation

$$\ln x - \ln x_0 \approx -\frac{\varepsilon_0 f_\varepsilon(x_0, \varepsilon_0)}{x_0 f_x(x_0, \varepsilon_0)} (\ln \varepsilon - \ln \varepsilon_0)$$

- Besides the fact that this produces expressions based on elasticities and share which economists like, there is no strong reason to prefer this to, for example, a linear approximation

3 Polynomial Approximation

- Suppose we have a function $f(x)$ that we are trying to approximate using polynomials

- Why polynomials?

(Weierstrass) If f is a continuous function that takes values in the $[a, b]$ interval, then for all $\varepsilon > 0$, there exists a polynomial $p(x)$ such that

$$\forall x \in [a, b],\ |f(x) - p(x)| \leq \varepsilon$$

This theorem tells us that for continuous functions there always exists a polynomial that (uniformly, which may be too strong a requirement for our purposes) approximates the function

- The importance is that it tells us that working with polynomial approximations may be a good idea

- We use basic vector space ideas to construct representations of functions that lead to good approximations (remember that a vector $x \in \mathbb{R}^n$ can be represented as a linear combination of n linearly independent vectors $\{v_1, v_2, ..., v_n\}$ of $\mathbb{R}^n$)

- Some more definitions:

- A **weighting function** $w(x)$, on $[a, b]$ is any function that is positive almost everywhere and has a finite integral on $[a, b]$. The inner product for a given $w(x)$ is

$$\langle f, g \rangle = \int_a^b f(x)g(x)w(x)dx$$

- The family of polynomials $\{\varphi_n(x)\}$ is **mutually orthogonal** with respect to the weighting function $w(x)$ if and only if $\langle \varphi_m, \varphi_n \rangle = 0$ for $m \neq n$.

- Actually, any polynomial can be expressed as a linear sum of orthogonal polynomials.

3.1 Least Squares Regression as Approximation

- We already know how to do a particular form of approximation. We know how to run linear regressions

- This is just a least squares approximation (i.e., it minimizes the sum of the squared errors).

- From data $\left\{y_i, x^i\right\}_{i=1}^n$ (for x^i of dimension k) we generate an approximation with $m < n$ functions $\phi_j : \mathbb{R}^k \to \mathbb{R}$ by solving

$$\min_{a_1,\ldots,a_m} \sum_{i=1}^{n} \left(a_1\phi_1\left(x^i\right) + \cdots + a_m\phi_m\left(x^i\right) - y_i\right)^2.$$

  This defines a function $\hat{f}(x) = \sum_{j=1}^{m} a_j\phi_j(x)$ such that it is close to y_i.

- A big difference is that in econometrics we are given the data so we cannot pick the "best" points. In numerical analysis we can pick the data! This makes all the difference and it means we may be able to do much better.

3.2 Least Square Orthogonal Polynomial Approximation

- Suppose we now choose some way of picking the polynomial

- One possible criteria is least squares

- Given some function $f(x)$ defined on $[a, b]$ the least squares polynomial approximation of f with respect to a weighting function $w(x)$ is the degree n polynomial that solves

$$\min_{\deg(p)\leq n} \int_a^b \left(f\left(x\right) - p\left(x\right)\right)^2 w\left(x\right) dx \tag{1}$$

  $w(x)$ indicates how much we care about approximation errors

- If $\{\varphi_n\left(x\right)\}_{n=0}^{\infty}$ is an orthogonal collection of polynomials with respect to a weighting function $w(x)$ on $[a, b]$ the solution to (1) can be expressed as

$$p(x) = \sum_{k=0}^{n} \frac{\langle f, \varphi_k \rangle}{\langle \varphi_k, \varphi_k \rangle} \varphi_k\left(x\right) \tag{2}$$

- This formula is essentially the same as regressing the function $f(x)$ on $n+1$ orthogonal regressors that are the $\varphi_k(x)$ functions. The coefficient of the k^{th} regressor, $\varphi_k(x)$, equals the "covariance" between f and the k^{th} regressor divided by the "variance" of the k^{th} regressor.

- Let's see the problem (1) from a slightly different angle. Write

$$p(x) = \sum_{i=0}^{n} \alpha_i \varphi_i(x)$$

and our goal is to find α_i, $i = 0, 1, ..., n$ such that (1) is solved.

- The first order conditions for this problem are

$$0 = 2\int_a^b w(x) \left[f(x) - \sum_{j=0}^{n} \alpha_j \varphi_j(x) \right] \varphi_i(x)\, dx, \ i = 0, 1, ..., n$$

which may be rewritten as

$$\int_a^b w(x) f(x) \varphi_i(x)\, dx = \sum_{j=0}^{n} \alpha_j \int_a^b w(x) \varphi_j(x) \varphi_i(x)\, dx, \ i = 0, 1, ..., n$$

- If the integral on the RHS of this equation vanishes for $i \neq j$ and equals a constant ζ_i for $i = j$, we have

$$\alpha_i = \frac{1}{\zeta_i} \int_a^b w(x) f(x) \varphi_i(x) dx.$$

- This motivates the following definition of orthogonal polynomials: A set of functions $\mathcal{P}$ is called orthogonal with respect to the weight function $w(x)$ if and only if

$$\int_a^b w(x) \varphi_j(x) \varphi_i(x) dx = \begin{cases} 0 & \text{if } i \neq j \\ \zeta_i & \text{if } i = j \end{cases}$$

- Compare this with (2)

Families of orthogonal polynomials

- We have several families of orthogonal polynomials to work with.

Family	$w(x)$	$[a, b]$	Definition $\varphi_i(x)$
Legendre	1	$[-1, 1]$	$P_n(x) = \frac{(-1)^n}{2^n n!} \frac{d^n[(1-x^2)^n]}{dx^n}$
Chebyshev	$\left(1 - x^2\right)^{-\frac{1}{2}}$	$[-1, 1]$	$T_n(x) = \cos(n \cos^{-1}(x))$
Laguerre	e^{-x}	$[0, \infty)$	$L_n(x) = \frac{e^x}{n!} \frac{d^n\left(x^n e^{-x}\right)}{dx^n}$
Hermite	e^{-x^2}	$(-\infty, \infty)$	$H_n(x) = (-1)^n e^{x^2} \frac{d^n\left(e^{-x^2}\right)}{dx^n}$

- The formulas in the table defining each family are too costly. It turns out we can define them recursively.

- First, note that $P_0(x) = T_0(x) = L_0(x) = H_0(x) = 1$ and $P_1(x) = T_1(x) = x$, $L_1(x) = 1 - x$ and $H_1(x) = 2x$.

- With this in hand

$$
\begin{aligned}
P_{n+1}(x) &= \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x) \\
L_{n+1}(x) &= \frac{1}{n+1} (2n + 1 - x) L_n(x) - \frac{n}{n+1} L_{n-1}(x) \\
T_{n+1}(x) &= 2x T_n(x) - T_{n-1}(x) \\
H_{n+1}(x) &= 2x H_n(x) - 2n H_{n-1}(x)
\end{aligned}
$$

- We investigate properties of Chebyshev polynomials in detail

3.3 Chebyshev Polynomials

- Chebyshev polynomials are defined on the $[-1, 1]$ interval by

$$T_n(x) = \cos(n \cos^{-1}(x))$$

and can be evaluated recursively as $T_0(x) = 1$, $T_1(x) = x$ and

$$T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x), \ i = 1, ..., n$$

  - For a proof of this recursive relation, see Heer and Maussner (2009), Ch 11.

- The following holds:

$$\int_{-1}^{1} \frac{T_i(x) T_j(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0 & i \neq j \\ \frac{\pi}{2} & i = j \neq 0 \\ \pi & i = j = 0 \end{cases}$$

- To approximate over $[a, b]$ instead of over $[-1, 1]$, we do a change of variables:

$$X(z) = \frac{2z - a - b}{b - a}, \quad z \in [a, b]$$

and

$$Z(x) = \frac{(x+1)(b-a)}{2} + a, \quad x \in [-1, 1].$$

- Define the function $p(Z(x))$ on the interval $[-1, 1]$ with approximation

$$p(z) = \sum_{i=0}^{n} \alpha_i T_i(X(z)).$$

Then, the coefficients of the continuous least squares approximation are

$$\alpha_i = \frac{2}{\pi} \int_a^b \frac{f(z) T_i(X(z))}{\sqrt{1 - (X(z))^2}} dz \tag{3}$$

- One way is to compute α_i in (3) by numerical integration. This is often difficult. Even if it is possible, the integrals cannot be evaluated exactly.

- Now we discuss methods that use a finite number of points to compute an approximation to $f(x)$. Two alternatives:

1. Approximations by interpolation. Use the same number of points as the degree of the approximating polynomial. Obtain the coefficients from a regression of $f(x)$ on x_i, $i = 1, ..., m$ where $m = n$.

2. Approximations by regression. Use more points than the degree of the approximating polynomial. Obtain the coefficients from a regression of $f(x)$ on x_i, $i = 1, ..., m$ where $m > n$.

- These can actually be used in practice

- First, let us state two convenient properties:

1. $T_n(x)$ has n zeros in the interval $[-1, 1]$ located at

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), \; k = 1, ..., n.$$

2. Discrete orthogonality: If $\{x_k\}_{k=1}^n$ are the n zeros of $T_n(x)$ and if $i, j < n$, then

$$\sum_{k=1}^{n} T_i(x_k) T_j(x_k) = \begin{cases} 0 & i \neq j \\ \frac{n}{2} & i = j \neq 0 \\ n & i = j = 0 \end{cases}$$

- Now consider the regression problem. In this case, we choose $m > n$ points as our data.

- In principle we can pick any points we want but we will again assume that we pick the zeros of the m^{th} degree polynomial.

- The reason to do this is simple. Remember we are trying to compute the degree $m-1$ interpolation formula but use only the degree n truncation in order to have a smoother function.

- Let $\bar{x} = [\bar{x}_1, ..., \bar{x}_m]$ denote the m zeros of $T_m(x)$.

- The corresponding points in the interval $[a, b]$ are $\bar{z}_k = Z(\bar{x}_k)$.

- We choose $\alpha = [\alpha_0, \alpha_1, ..., \alpha_{n-1}]$ to minimize

$$\sum_{k=1}^{m} \left[f(\bar{z}_k) - \sum_{j=1}^{n-1} \alpha_j T_j (X(\bar{z}_k)) \right].$$

The first order conditions are

$$\sum_{k=1}^{m} T_0 f(\bar{z}_k) = \sum_{j=0}^{n-1} \alpha_j \sum_{k=1}^{n} T_0 T_j(\bar{x}_k) = m\alpha_0$$

$$\sum_{k=1}^{m} f(\bar{z}_k) T_1(\bar{x}_k) = \sum_{j=0}^{n-1} \alpha_j \sum_{k=1}^{m} T_1(\bar{x}_k) T_j(\bar{x}_k) = (m/2)\alpha_1$$

$$\vdots$$

$$\sum_{k=1}^{m} f(\bar{z}_k) T_{n-1}(\bar{x}_k) = \sum_{j=0}^{n-1} \alpha_j \sum_{k=1}^{m} T_{n-1}(\bar{x}_k) T_j(\bar{x}_k) = (m/2)\alpha_{n-1}$$

- The last equality of each equation holds because of the discrete orthogonality. (definition of $\bar{x}$ and $m > n$ are important)

- Chebyshev Regression Algorithm: choose m nodes and use them to construct a degree $n < m$ polynomial approximation of $f(x)$ on $[a, b]$.

  1. Compute the $m \geq n$ interpolation nodes on $[-1, 1]$

$$\bar{x}_k = \cos\left(\frac{2k-1}{2m}\pi\right), \ k = 1, ..., m$$

  2. Adjust the nodes to $[a, b]$

$$\bar{z}_k = (\bar{x}_k + 1)\left(\frac{b-a}{2}\right) + a, \ k = 1, ..., m$$

  3. Evaluate f at the nodes

$$w_k = f\left(\bar{z}_k\right), \ k = 1, ..., m$$

4. Compute the n coefficients a_i, $i = 0, ..., n-1$

$$\alpha_i = \frac{\sum_{k=1}^{m} w_k T_i(\bar{x}_k)}{\sum_{k=1}^{m} T_i(\bar{x}_k)^2}.$$

5. Approximate f

$$\hat{f}(x) = \sum_{i=0}^{n-1} \alpha_i T_i\left(2\frac{x-a}{b-a} - 1\right)$$

- We next consider the Chebyshev polynomial approximation by interpolation (assuming we approximate f in the interval $[-1, 1]$ for simplicity)

- If we let $\bar{x} = [\bar{x}_1, ..., \bar{x}_n]$ be the n zeros of the n^{th} degree Chebyshev polynomial and we define n coefficients

$$\alpha_0 = \frac{\sum_{k=1}^{n} f(\bar{x}_k)}{n} \tag{4}$$

$$\alpha_i = \frac{2\sum_{k=1}^{n} f(\bar{x}_k) T_i(\bar{x}_k)}{n} \text{ for } i = 1, ..., n-1 \tag{5}$$

then the approximation formula

$$f(x) \approx \sum_{i=0}^{n-1} a_i T_i(x)$$

is *exact* for x equal to the n zeros of $T_n(x)$.

- We can actually prove that interpolation at the zeros of Chebyshev polynomials leads to asymptotically valid approximations of f for C^1 smooth functions.

- We can also show that interpolation with Chebyshev polynomials has the minimax property so it has the smallest possible maximum interpolation error.

- All of these make approximation using Chebyshev polynomials very attractive.

3.4 Application: Dynamic Discrete Choice

- A dynamic discrete-choice problem

  - Decision periods: $t = 1, ..., T < \infty$ (e.g., life cycle model)
  - Values and period utilities have a time subscript.
  - Decision set, $D(x_t)$: a finite set of allowable values of the control variable i_t when state variable is x_t.
  - Unobserved state variables, $\varepsilon_t = \{\varepsilon_t(i) | i \in D(x_t)\}$
  - Observed state variables, $x_t = \{x_t(1), ..., x_t(K)\}$
  - Immediate reward function, $u_t(x_t, i; \theta) + \varepsilon_t(i)$
  - Conditional Independence (CI) assumption:

$$p_t(x_{t+1}, \varepsilon_{t+1} | x_t, \varepsilon_t, i_t, \theta) = q_t(\varepsilon_{t+1} | x_{t+1}, \theta) p_t(x_{t+1} | x_t, i_t, \theta)$$

- The value function

$$V_t(x_t, \varepsilon_t, \theta) = \max_k \left\{ \bar{V}_t^k(x_t, \theta) + \varepsilon_{kt} \right\}$$

where $\bar{V}^k(x_t, \theta)$ is the alternative-specific value function, given by

$$\bar{V}_t^k(x_t, \theta) = u_t(x_t, k, \theta) + \beta E[V_{t+1}(x_{t+1}, \varepsilon_{t+1}, \theta) | x_t, i_t = k, \theta]$$

for $t \leq T - 1$ and

$$\bar{V}_T^k(x_T, \theta) = u_T(x_T, k, \theta).$$

- Then,

$$\bar{V}_t^k(x_t, \theta) = u_t(x_t, k, \theta) + \beta E[\max_{i_{t+1}} \left\{ \begin{array}{c} \bar{V}_{t+1}^1(x_{t+1}, \theta) + \varepsilon_{1t+1}, \\ \vdots \\ \bar{V}_{t+1}^K(x_{t+1}, \theta) + \varepsilon_{Kt+1} \end{array} \right\} | x_t, k, \theta],$$

for $t \leq T - 1$ and

$$\bar{V}_T^k(x_T, \theta) = u_T(x_T, k, \theta).$$

- Call the continuation value $\text{EMAX}_t(x_t)$.

- If we are willing to assume iid Type 1 extreme values for ε_t, $\text{EMAX}_t(x_t)$ has the closed form:

$$\bar{V}_t^k(x_t, \theta) = u_t(x_t, k, \theta) + \beta E[\mu + \ln \left\{ \sum_j \exp(\bar{V}_{t+1}^j(x_{t+1}, \theta)) \right\} | x_t, k, \theta],$$

  where μ is called the Euler constant. This is what we did in our discussion of the Rust model. (We did not have a nonstationary problem, so this relationship could already be used for the value function iteration)

- In addition, the extreme-value assumption gives the closed form for choice probabilities:

$$\Pr(i_t = k | x_t, t, \theta) = \frac{\exp(\bar{V}_t^k(x_t, \theta))}{\sum_j \exp(\bar{V}_t^j(x_t, \theta))}.$$

- But what if we do not want to assume the extreme value distribution for ε? What if we want to introduce correlation in ε between alternatives?

- Keane and Wolpin (1994) use simulation methods to account for such cases. In addition, they use interpolation to deal with large state spaces.

- Let $E\bar{V}_t^k(x_t, \theta)$ be the expected value of $\bar{V}_t^k(x_t, \theta)$ and let $\text{MAXE}_t(x_t) \equiv \max_k \left\{ E\bar{V}_t^k(x_t, \theta) \right\}$

- Approximate EMAX_t

$$\text{EMAX}_t\left(x_t\right) \approx \text{MAXE}_t\left(x_t\right) + g_t\left(\left\{\text{MAXE}_t\left(x_t\right) - \bar{V}_t^k(x_t, \theta)\right\}_{k=1}^{K}\right) \quad (6)$$

  where g_t takes $\text{MAXE}_t\left(x_t\right) - \bar{V}_t^k(x_t, \theta)$ for all k as arguments.

- $g() > 0$. Why?

- Backwards recursive solution

  - At T, simulate EMAX$_T$ using Monte Carlo (use pseudo random draws ε_t for each time) for a subset of the entire state space, denoted by $S^*(T)$. This is easy because there are no continuation payoffs at T

  - These simulated EMAX along with $\bar{V}_t^k(x_t, \theta)$ are used to "estimate" g_t.

  - Moving backwards, at $T-1$, we want to compute EMAX$_{T-1}$ for $S^*(T-1)$.

  - To do so, we need EMAX$_t$ for points in $S^*(T)$ and all points that can be reached from a point in $S^*(T-1)$.

  - If those points are in $S^*(T)$, we already have those simulated. For those not in $S^*(T)$ we use (6) to interpolate them and get fitted values for all relevant states

  - Repeat this procedure backwards until $t = 1$

- Based on extensive simulation exercises, Keane and Wolpin (1994) recommend the following functional form for g :

$$
\begin{aligned}
& g_t\left(\left\{\mathrm{MAXE}_t\left(x_t\right)-\bar{V}_t^k(x_t,\theta)\right\}_{k=1}^K\right) \\
& =\pi_0+\sum_{j=1}^K \pi_{1j}\left(\mathrm{MAXE}_t\left(x_t\right)-\bar{V}_t^j(x_t,\theta)\right) \\
& +\sum_{j=1}^K \pi_{2j}\left(\mathrm{MAXE}_t\left(x_t\right)-\bar{V}_t^j(x_t,\theta)\right)^{1/2}
\end{aligned}
$$

where π's are time varying. Estimate this by OLS.

- No theory behind this specification, but easy to implement so widely used in application.

3.5 Piecewise Polynomial Interpolation

- Simplest way is to connect the dots. With data $\{x_i, y_i\}$

$$\hat{f}(x) = y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i) \quad x \in [x_i, x_{i+1}]$$

- Notice this is a simple form of a piecewise linear interpolator (i.e., a particular form of piecewise polynomial interpolation, linear in this case)

- Linear interpolation is simple and shape preserving. This is important, if we use interpolation to approximate the value function, which is known to be concave and increasing.

- **Splines**:

- Splines are a form of piecewise polynomial interpolation that is really popular.

- Spline: any smooth function that is piecewise polynomial AND smooth where the polynomial pieces connect.

- $s(x)$ is a spline of order n on $[a, b]$ if s is C^{n-2} and there are nodes $a = x_0 < x_1 < \cdots < x_m = b$ such that $s(x)$ is a polynomial of degree $n - 1$ on each subinterval $[x_i, x_{i+1}]$ for $i = 0, 1, ..., m - 1$.

- Piecewise linear interpolation is the spline of order 2.

- The most common form of splines are **cubic splines** (i.e., splines of order 4)

- Suppose we have data $\{x_i, y_i\}_{i=0}^{n}$. x_i are the nodes of the spline.

- We want $s\,(x)$ such that $s\,(x_i) = y_i$ for x_i a node and that, for $x \in [x_i, x_{i+1}]$, we have $s\,(x) = a_i + b_i x + c_i x^2 + d_i x^3$.

- Thus, there are $4n$ coefficients we want to determine

- The interpolating conditions and the continuity (at interior nodes) conditions imply $2n$ restrictions:

$$y_i = a_i + b_i x_i + c_i x_i^2 + d_i x_i^3, \quad i = 1, ..., n$$
$$y_i = a_{i+1} + b_{i+1} x_i + c_{i+1} x_i^2 + d_{i+1} x_i^3, \quad i = 0, ..., n-1$$

- We also impose that the approximation is C^2 so first and second derivatives should coincide (on interior nodes): we have $2n-2$ more conditions

$$b_i + 2c_i x_i + 3d_i x_i^2 = b_{i+1} + 2c_{i+1} x_i + 3d_{i+1} x_i^2, \quad i = 1, ..., n-1$$
$$2c_i + 6d_i x_i = 2c_{i+1} + 6d_{i+1} x_i, \quad i = 1, ..., n-1.$$

- How to fill two missing conditions? There are several ways:

1. Natural splines impose $s'(x_0) = s'(x_n) = 0$

2. Maybe we have a good guess for y'_0 and y'_n.

3. We can use the slope of the secants $s'(x_0) = \frac{s(x_1)-s(x_0)}{x_1-x_0}$ and $s'(x_n) = \frac{s(x_n)-s(x_{n-1})}{x_n-x_{n-1}}$.

4. Other possibilities exist

- We like splines because they are cheap and because they approximate well in general

4 Chebyshev Approximations for Functions of more than one variable

- Finally we look at approximating functions of more than one variable. We start by defining a tensor product basis and a complete polynomial basis for the three dimensional case.

- If A and B are sets of functions over $x \in \mathbb{R}^a$, $y \in \mathbb{R}^b$, their tensor product is

$$\mathbf{\Phi} = \{\varphi(x)\,\psi(y)\,|\varphi \in A, \psi \in B\}\,.$$

- Now, given a basis for functions of one variable x_i, $\mathbf{\Phi}^i = \left\{\varphi_k^i(x_i)\right\}_{k=0}^{\infty}$ we can build the n-fold tensor product basis for functions of n variables $(x_1 ..., x_n)$ by taking all possible n-term products of φ_k^i.

- The resulting basis is

$$\Phi = \left\{ \prod_{i=1}^{n} \varphi_k^i (x_i) \,|k = 0, ..., i = 1, ..., n \right\}$$

- We will only use a finite subset of the full tensor product basis. Say, the first m_i elements for each univariate basis and construct its tensor product

- Tensor products will allow us to adapt one dimensional methods to higher dimensions.

- Take the following Chebyshev approximation algorithm for the case of a function of 3 variables, $f(x, y, z)$, defined on $[a_x, b_x] \times [a_y, b_y] \times [a_z, b_z]$ using degree $n - 1$ polynomials for each variable (no reason why it should be the same degree for all variables):

1. Compute the $m \geq n$ interpolation nodes on $[-1, 1]$

$$\gamma_k = -\cos\left(\frac{2k-1}{2m}\pi\right),\ k = 1, ..., m.$$

2. Adjust the nodes to $[a_i, b_i]$

$$x_k = (\gamma_k + 1)\left(\frac{b_x - a_x}{2}\right) + a_x,\ k = 1, ..., m,$$
$$y_k = (\gamma_k + 1)\left(\frac{b_y - a_y}{2}\right) + a_y,\ k = 1, ..., m,$$
$$z_k = (\gamma_k + 1)\left(\frac{b_z - a_z}{2}\right) + a_z,\ k = 1, ..., m.$$

3. Evaluate f at the nodes

$$w_{g,h,i} = f\left(x_g, y_h, z_i\right),\ g, h, i \in \{1, ..., m\}$$

4. Compute the n^3 coefficients $a_{j,k,l}$, $j, k, l \in \{0, ..., n-1\}$

$$a_{j,k,l} = \frac{\sum_{g=1}^{m}\sum_{h=1}^{m}\sum_{i=1}^{m} w_{g,h,i} T_j\left(\gamma_g\right) T_k\left(\gamma_h\right) T_l\left(\gamma_i\right)}{\left(\sum_{g=1}^{m} T_j\left(\gamma_g\right)^2\right)\left(\sum_{h=1}^{m} T_k\left(\gamma_h\right)^2\right)\left(\sum_{i=1}^{m} T_l\left(\gamma_i\right)^2\right)}$$

5. Approximate f using tensor products

$$\hat{f}(x,y,z) = \sum_{j=0}^{n-1}\sum_{k=0}^{n-1}\sum_{l=0}^{n-1} a_{j,k,l} T_j\left(2\frac{x-a_x}{b_x-a_x}-1\right) \times T_k\left(2\frac{y-a_y}{b_y-a_y}-1\right) T_l\left(2\frac{z-a_z}{b_z-a_z}-1\right)$$

- Notice that if we have a d-dimensional function that we wish to approximate with degree $n-1$ polynomials, we will have n^d terms in the approximating function

- This exponential growth of the number of terms for the approximation is known as the curse of dimensionality

- To avoid this problem we can use **complete polynomial basis** since they only grow polynomially

- Suppose we have three variables x_1, x_2, x_3. Their tensor product would include all possible interactions of this variables:

- To see how complete polynomials help, take the approximating function in (5) above.

- Define T_j^x as the degree j Chebyshev polynomial evaluated at x and so on, so that we can rewrite (5) as

$$\hat{f}(x, y, z) = \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} a_{j,k,l} T_j^x T_k^y T_l^z$$

- If, for example, we use degree 2 polynomials in (5) (so $n = 3$) the above sum would have 27 (n^3) terms

- The same formula based on a complete polynomial basis of the same degree however would only contain 10 terms $\left(\frac{n-1+d!}{n-1!d!}\right)$ terms since, if we define the index

set $\Omega = \{i, j, k | i + j + k \leq n - 1\}$ that defines the complete polynomial of degree $n - 1$ for this case, we can write (5) based on complete polynomials as:

$$\hat{f}(x, y, z) = \sum_{j,k,l \in \Omega}^{n-1} a_{j,k,l} T_j^x T_k^y T_l^z$$

- It may seem like a bad idea to drop so many terms but, in terms of asymptotic convergence, complete polynomial approximations will yield the same rate ($n-1$) as the tensor product approximation.

- The reason is Taylor's theorem.

- We use complete polynomials of final degree in the hope that the asymptotics kick in at a practical degree.