Function Approximation and Functional Equations

Roberto Chang

Rutgers

April 2013
Recall that the solution of the Lucas asset price model is given by:

\[ p(z)u'(z) = \beta \int u'(z')[z' + p(z')]Q(z, dz') \]

This is a functional equation where the unknown is a function \( p : Z \rightarrow \mathbb{R}_+ \)

Suppose that you know \( u, Z, Q, \beta \). How do you compute an approximate solution?
Numerical Issues

At least two nontrivial issues appear if e.g. $Z$ is an interval:

1. **How do you compute the integral?**

In this case, both issues "disappear" if one assumes that $Z$ is a finite set. But in other contexts that may be unnatural or misleading. Hence we review procedures to deal squarely with the two issues. Additionally, we discuss a computational procedure to solve functional equations.
At least two nontrivial issues appear if e.g. $Z$ is an interval:

1. How do you compute the integral?
2. How do you even represent or approximate a candidate solution, a function $p : Z \to \mathbb{R}_{++}$?

In this case, both issues "disappear" if one assumes that $Z$ is a finite set. But in other contexts that may be unnatural or misleading. Hence we review procedures to deal squarely with the two issues. Additionally, we discuss a computational procedure to solve functional equations.
The problem is to approximate a (possibly intractable) real valued function $f$ with a computationally tractable $\hat{f}$, using only limited information about $f$. 
The problem is to approximate a (possibly intractable) real valued function $f$ with a computationally tractable $\hat{f}$, using only limited information about $f$.

This turns out to be extremely useful in many contexts. For example, if one iterates on Bellman:

$$v^{(i+1)}(k, z) = \max_a u(k, z, a) + \beta \int v^{(i)}(k', z') Q(z, dz')$$

s.t. $a \in \Gamma(k, z)$

$$k' = \phi(k, a, z')$$

at each step one only needs to solve for each iteration $v^{(i+1)}$ at a finite set of values in $K \times Z$, then form the approximation $\hat{v}^{(i+1)}$. 
A useful approach to approximate a function \( f \) is to choose an approximant from a given family:

\[
\hat{f}(x) = \sum_{j=1}^{n} c_j \phi_j(x)
\]

where \( \phi_j(x), j = 1, \ldots, n \) are known \textit{basis functions} and \( c_1, \ldots, c_n \) are coefficients that pin down an approximant. \( n \) is called the \textit{degree} of the approximation.
A useful approach to approximate a function \( f \) is to choose an approximant from a given family:

\[
\hat{f}(x) = \sum_{j=1}^{n} c_j \phi_j(x)
\]

where \( \phi_j(x), j = 1, \ldots, n \) are known basis functions and \( c_1, \ldots, c_n \) are coefficients that pin down an approximant. \( n \) is called the degree of the approximation.

Obvious monomial example: let \( \phi_j(x) = x^{j-1} \), so an \( n^{th} \) degree approximation to \( f \) is the polynomial \( c_0 + c_1 x + \ldots + c_n x^{n-1} \).
Approximation Decisions:

- How do you choose basis functions?
Approximation Decisions:

- How do you choose basis functions?
- Given basis functions, how do you choose the coefficients of the approximation?
Choosing the Coefficients: Interpolation

Let us focus on the choice of coefficients first: suppose that we are given a family of basis functions.
Let us focus on the choice of coefficients first: suppose that we are given a family of basis functions.

A typical situation: we know the values of $f$ at some $n$ nodes $x_1, ..., x_n$:

$f(x_k) = y_k, k = 1, ..., n$. 

This is a linear system of $n$ equations in the $n$ unknown coefficients $c_1, ..., c_n$.

Often we can choose the nodes, so this is another decision to be made.
Choosing the Coefficients: Interpolation

- Let us focus on the choice of coefficients first: suppose that we are given a family of basis functions.
- A typical situation: we know the values of $f$ at some $n$ nodes $x_1, ..., x_n : f(x_k) = y_k, k = 1, ..., n$.
- Then we can choose the nodes by solving:

$$\hat{f}(x_k) = \sum_{j=1}^{n} c_j \phi_j(x_k) = y_k$$
Let us focus on the choice of coefficients first: suppose that we are given a family of basis functions.

A typical situation: we know the values of $f$ at some $n$ nodes $x_1, \ldots, x_n$: $f(x_k) = y_k$, $k = 1, \ldots, n$.

Then we can choose the nodes by solving:

$$\hat{f}(x_k) = \sum_{j=1}^{n} c_j \phi_j(x_k) = y_k$$

This is a linear system of $n$ equations in the $n$ unknown coefficients $c_1, \ldots, c_n$.
Choosing the Coefficients: Interpolation

- Let us focus on the choice of coefficients first: suppose that we are given a family of basis functions.
- A typical situation: we know the values of $f$ at some $n$ nodes $x_1, ..., x_n:\ f(x_k) = y_k, k = 1, ..., n$.
- Then we can choose the nodes by solving:

$$\hat{f}(x_k) = \sum_{j=1}^{n} c_j \phi_j(x_k) = y_k$$

- This is a linear system of $n$ equations in the $n$ unknown coefficients $c_1, ..., c_n$
- Often we can choose the nodes, so this is another decision to be made.
Choosing Basis Functions: Spectral and Finite Element Methods

- Spectral methods use basis functions that are nonzero at almost all points of the domain (e.g. monomials). The most popular such method is polynomial interpolation.
Choosing Basis Functions: Spectral and Finite Element Methods

- Spectral methods use basis functions that are nonzero at almost all points of the domain (e.g. monomials). The most popular such method is polynomial interpolation.
- Finite element methods uses basis functions that are nonzero only over subintervals of the domain. Most popular: linear and cubic splines.
The obvious choice for a basis, monomials, is not good because they are far from orthogonal as $n$ increases. This makes the interpolation system ill conditioned.
The obvious choice for a basis, monomials, is not good because they are far from orthogonal as \( n \) increases. This makes the interpolation system ill conditioned.

A much better alternative: Chebychev polynomials. For \( x \) in \([a, b]\), let \( z = (x - a)/(b - a) \) and define:

\[
T_0(z) = 1, \quad T_1(z) = z \\
T_j(z) = 2zT_{j-1}(z) - T_{j-2}(z), j \geq 2
\]
With polynomial approximation, the obvious choice of nodes is equidistant.
Choice of Nodes

- With polynomial approximation, the obvious choice of nodes is equidistant.
- But this is known to be problematic.
Choice of Nodes

- With polynomial approximation, the obvious choice of nodes is equidistant.
- But this is known to be problematic.
- Preferred: Chebychev nodes, for $i = 1, \ldots, n$

$$x_i = \frac{a + b}{2} + \frac{b - a}{2} \cos \left( \frac{n - i + 0.5}{n} \pi \right)$$
Instead of constructing a high degree smooth approximation that interpolates the data, one can subdivide the domain and fit low order polynomials to each part of the domain.
Instead of constructing a high degree smooth approximation that interpolates the data, one can subdivide the domain and fit low order polynomials to each part of the domain.

The simplest method is *piecewise linear approximation*, aka "connect the dots "

Piecewise Polynomial Interpolation
Instead of constructing a high degree smooth approximation that interpolates the data, one can subdivide the domain and fit low order polynomials to each part of the domain.

The simplest method is *piecewise linear approximation*, aka "connect the dots"

A popular alternative: piecewise cubic
Instead of constructing a high degree smooth approximation that interpolates the data, one can subdivide the domain and fit low order polynomials to each part of the domain.

The simplest method is \textit{piecewise linear approximation}, aka "connect the dots."

A popular alternative: piecewise cubic

Piecewise polynomial approximations can be seen as linear combination of basis functions called \textit{splines}.
A spline of order $n$ on $[a, b]$ is a function $s : [a, b] \rightarrow \mathbb{R}$ such that:

- $s$ has continuous derivatives up to order $n - 2$
A spline of order $n$ on $[a, b]$ is a function $s : [a, b] \rightarrow \mathbb{R}$ such that:

- $s$ has continuous derivatives up to order $n - 2$
- There are $m$ nodes $a = x_1 < ... < x_m = b$ such that $s(x)$ is a polynomial of degree $n - 1$ on each subinterval $[x_i, x_{i+1}]$
A spline of order $n$ on $[a, b]$ is a function $s : [a, b] \to \mathbb{R}$ such that:

- $s$ has continuous derivatives up to order $n - 2$
- There are $m$ nodes $a = x_1 < \ldots < x_m = b$ such that $s(x)$ is a polynomial of degree $n - 1$ on each subinterval $[x_i, x_{i+1}]$
- An order 2 spline is just the common linear interpolant
Suppose we have $n + 1$ nodes, $x_0, x_1, \ldots, x_n$, and we know $y_i = f(x_i)$ at each node. We want to construct an interpolating cubic spline on each subinterval $[x_i, x_{i+1}]$, the spline will have the representation $a_i + b_i x + c_i x^2 + d_i x^3$. Hence we have to find $4n$ coefficients.

The interpolation conditions, plus continuity and smoothness at the interior points, give $4n - 2$ conditions. The two extra conditions are solved in different ways (natural, Hermite, not-a-knot). The resulting system of equations to be solved is often linear and sparse.
Suppose we have \( n + 1 \) nodes, \( x_0, x_1 \ldots x_n \), and we know \( y_i = f(x_i) \) at each node. We want to construct an interpolating cubic spline.

On each subinterval \([x_i, x_{i+1}]\), the spline will have the representation:

\[
a_i + b_i x + c_i x^2 + d_i x^3
\]

Hence we have to solve for 4\( n \) coefficients.

The interpolation conditions, plus continuity and smoothness at the interior points, give 4\( n \) - 2 conditions. The two extra conditions are solved in different ways (natural, Hermite, not-a-knot). The resulting system of equations to be solved is often linear and sparse.
Suppose we have \( n + 1 \) nodes, \( x_0, x_1 \ldots x_n \), and we know \( y_i = f(x_i) \) at each node. We want to construct an interpolating cubic spline.

On each subinterval \([x_i, x_{i+1}]\), the spline will have the representation \( a_i + b_ix + c_ix^2 + d_ix^3 \).

Hence we have to fix \( 4n \) coefficients.
Suppose we have $n + 1$ nodes, $x_0, x_1 \ldots x_n$, and we know $y_i = f(x_i)$ at each node. We want to construct an interpolating cubic spline.

On each subinterval $[x_i, x_{i+1}]$, the spline will have the representation $a_i + b_i x + c_i x^2 + d_i x^3$.

Hence we have to fix $4n$ coefficients.

The interpolation conditions, plus continuity and smoothness at the interior points, give $4n - 2$ conditions.
Suppose we have \( n + 1 \) nodes, \( x_0, x_1 \ldots x_n \), and we know \( y_i = f(x_i) \) at each node. We want to construct an interpolating cubic spline.

On each subinterval \([x_i, x_{i+1}]\), the spline will have the representation
\[
a_i + b_i x + c_i x^2 + d_i x^3
\]

Hence we have to fix \( 4n \) coefficients.

The interpolation conditions, plus continuity and smoothness at the interior points, give \( 4n - 2 \) conditions.

The two extra conditions are solved in different ways (natural, Hermite, not-a-knot).
Cubic Splines

- Suppose we have \( n + 1 \) nodes, \( x_0, x_1, \ldots, x_n \), and we know \( y_i = f(x_i) \) at each node. We want to construct an interpolating cubic spline.
- On each subinterval \([x_i, x_{i+1}]\), the spline will have the representation \( a_i + b_i x + c_i x^2 + d_i x^3 \).
- Hence we have to fix \( 4n \) coefficients.
- The interpolation conditions, plus continuity and smoothness at the interior points, give \( 4n - 2 \) conditions.
- The two extra conditions are solved in different ways (natural, Hermite, not-a-knot).
- The resulting system of equations to be solved is often linear and sparse.
As mentioned, splines can be expressed as linear combinations of a basis family called B-splines.
As mentioned, splines can be expressed as linear combinations of a basis family called B-splines.

For piecewise linear splines, B-splines are "tent functions ".

Basis for Splines: B-splines

- As mentioned, splines can be expressed as linear combinations of a basis family called B-splines.
- For piecewise linear splines, B-splines are "tent functions".
- For cubic splines and others, see Judd or MF.
Keep in mind that spectral methods fit a global approximant, whereas finite element methods are "local."
Choosing an Approximation Method

- Keep in mind that spectral methods fit a global approximant, whereas finite element methods are "local."
- For smooth functions, polynomial approximations are very good.
Choosing an Approximation Method

- Keep in mind that spectral methods fit a global approximant, whereas finite element methods are "local"
- For smooth functions, polynomial approximations are very good
- If one has discontinuities, kinks, etc. splines may be preferable
Consider approximating $f(x, y)$. If $\{\phi_i(x)\}_{i=1}^n$ and $\{\eta_j(y)\}_{j=1}^m$ are one dimensional basis families, a basis family for the two dimensional case is given by the tensor family of products $\phi_i(x)\eta_j(y)$.

Likewise, if $\{x_1...x_n\}$ and $\{y_1...y_m\}$ are nodes in the unidimensional case, for the two dimensional case one can use the nodes $\{(x_i, y_j)\}$.
global vlast betta del theta k0 kt
vlast = zeros(1,100);
k0 = 0.06:0.06:6;
betta = 0.98; del = 0.1; theta = 0.36; numits = 240;
for k = 1:numits;
    for j = 1:100
        kt = j * 0.06;
        ktp1 = fminbnd(@valfun,0.01,6.2);
        v(j) = -valfun(ktp1);
        kt1(j) = ktp1;
    end
    vlast = v;
end
function val = valfun(x)
%VALFUN From Mc Candless, p. 67
% Auxiliary function

global vlast betta del theta k0 kt

cc = kt^theta + (1 - del)* kt - x;
g = interp1(k0, vlast, x, 'spline');

if cc<=0
    val = -888 - 800*abs(cc);
else
    val = log(cc) + betta*g;
end

val = -val;
end
Consider the problem: find a function $f : D \rightarrow \mathbb{R}$, $f \in F$, such that for all $x \in D$

$$Tf(x) = 0$$

where $T : F \rightarrow F$ is an operator on $F$. 

Example: rewrite the Lucas tree problem as

$$T_p(z) = p(z)u_0(z)\left[z_0 + p(z_0)Q(z, dz_0)\right] = 0$$

More generally: problems whose solutions are given by systems of functional equations.
Consider the problem: find a function $f : D \rightarrow \mathbb{R}$, $f \in F$, such that for all $x \in D$

$$Tf(x) = 0$$

where $T : F \rightarrow F$ is an operator on $F$

Example: rewrite the Lucas tree problem as

$$Tp(z) \equiv p(z)u'(z) - \beta \int u'(z')[z' + p(z')]Q(z, dz') = 0$$
Consider the problem: find a function \( f : D \rightarrow \mathbb{R} \), \( f \in F \), such that for all \( x \in D \)

\[
Tf(x) = 0
\]

where \( T : F \rightarrow F \) is an operator on \( F \)

Example: rewrite the Lucas tree problem as

\[
Tp(z) \equiv p(z)u'(z) - \beta \int u'(z')[z' + p(z')]Q(z, dz') = 0
\]

More generally: problems whose solutions are given by systems of functional equations
Suppose that we will look for an approximate solution in the family

\[ \hat{f}(x; c) = \sum_{j=1}^{n} c_j \phi_j(x) \]
Suppose that we will look for an approximate solution in the family

\[ \hat{f}(x; c) = \sum_{j=1}^{n} c_j \phi_j(x) \]

Fix the degree of the approximation, \( n \). Then the collocation method requires the functional equation to hold exactly at \( n \) points (nodes) in the domain:

\[ T \hat{f}(x_i; c) = T(\sum_{j=1}^{n} c_j \phi_j)(x_i) = 0, \quad i = 1, \ldots, n \]
Suppose that we will look for an approximate solution in the family

\[ \hat{f}(x; c) = \sum_{j=1}^{n} c_j \phi_j(x) \]

Fix the degree of the approximation, \( n \). Then the collocation method requires the functional equation to hold exactly at \( n \) points (nodes) in the domain:

\[ T \hat{f}(x_i; c) = T(\sum_{j=1}^{n} c_j \phi_j)(x_i) = 0, \quad i = 1, \ldots, n \]

This gives a (probably nonlinear) system of \( n \) equations for the \( n \) unknown coefficients \( c_1 \ldots c_n \)
Away from the nodes, the *residual function*:

\[ R(x; c) = T\hat{f}(x; c) = T\left(\sum_{j=1}^{n} c_j \phi_j\right)(x) \]

will not be zero. The quality of the approximation can be judged by looking at the residual function.
Away from the nodes, the *residual function*:

\[ R(x; c) = T\hat{f}(x; c) = T\left(\sum_{j=1}^{n} c_j \phi_j\right)(x) \]

will not be zero. The quality of the approximation can be judged by looking at the residual function.

Other methods choose \( c \) to make the residual function close to zero in different ways, e.g. on average.
Away from the nodes, the \textit{residual function}:

\[ R(x; c) = T \hat{f}(x; c) = T \left( \sum_{j=1}^{n} c_j \phi_j \right)(x) \]

will not be zero. The quality of the approximation can be judged by looking at the residual function.

Other methods choose \( c \) to make the residual function close to zero in different ways, e.g. on average.

For example, one could choose \( c_1 \ldots c_n \) to minimize a version of least squares:

\[ \int_{a}^{b} [R(x; c)]^2 w(x) \, dx \]

for some weight function \( w \).