# Dynamic optimization

Felix Kubler[1]

[1]DBF, University of Zurich and Swiss Finance Institute

July 4, 2019

# Dynamic economies

- Time $t = 0, 1, \ldots$.
- Uncertainty: Each period a shock $s_t \in \mathcal{S} = \{1, \ldots, S\}$ realizes. $(s_t)$ follows a Markov chain with transition $\pi$
- Denote a history of shocks by $s^t = (s_0, \ldots, s_t)$ – this can be associated with a node in the infinite event tree
- Economy is stationary if endowments and technology just a function of current shock.

# Infinitely lived agents

- Assume that at each node, there is a single perishable consumption good. Agents live forever.
- There exist some constant $\beta$, probabilities $\pi(\sigma) = \pi(\sigma|\sigma_0)$ and a function $v : \mathbb{R}_+ \to \mathbb{R}$ such that

$$U(c) = \sum_{t=0}^{\infty} \beta^t \sum_{\sigma \in \mathcal{N}_t} \pi(\sigma) v(c(\sigma))$$

# The Stochastic Ramsey Model

$$\max \mathbb{E}_0[U(\{c_t\})] \quad \text{s.t. } c_t + k_{t+1} \leq \underbrace{z(s_t)f(k_t) + (1-\delta(s_t))k_t}_{\equiv \bar{f}(s_t,k_t)},$$

$$c_t \geq 0,\ k_{t+1} \geq 0\ \forall t \in \mathbb{N}_0$$

$$k_0, s_0 \text{ given}$$

where the expectation is over the sequence of stocks $\{s_t\}_{t=1}^{\infty}$ given $s_0$.

Special Cases:

- $\delta$ fixed and AR(1)-process for TFP shock
  $\ln z_{t+1} = \rho \ln z_t + \epsilon_{t+1}$ where $\epsilon_{t+1} \sim \mathcal{N}(0, \sigma_\epsilon)$
- $\delta$ fixed and discrete Markov process:
  $z_t \in \mathcal{Z} = \{1, \ldots, Z\}, \mathbb{P}(z_{t+1} = i | z_t = j) = \pi_{ij}$
  with transition matrix $\pi$

# The Stochastic Ramsey Model

- How can we solve the stochastic model?
- Want consumption, investment as a function of the shock and the capital-stock
- Can use local methods, *perturbation methods*, that are variations on Taylor's theorem
- Want to advocate global methods.
- In this simple case can use numerical dynamic programming...

# Markov Control Processes

Fancy treatment of dynamic programming. See e.g. Hernandez-Lerma and Lasserre 1996 , “Discrete-Time Markov Control Processes”

or, at least, Stokey and Lucas w. Prescott.

# Markov Control Processes

Suppose the state space is $X \subset \mathbb{R}^l$ with Borel $\sigma$-algebra. Actions are $A : X \rightrightarrows \mathbb{R}^k$. State transition is $Q(.|x, a)$ The objective is

$$V(\pi, x) = E_x^\pi \sum_{t=0}^{\infty} \beta^t c(x_t, a_t), x \in X, \pi \in \Pi$$

Want to find optimal policy $\pi^*$ such that

$$V(\pi^*, x) = \inf_\pi V(\pi, x) =: V^*(x)$$

Typically do not want to allow for randomizations, i.e. can associate an optimal policy with $(a_t)$.
A measurable function $v : X \to \mathbb{R}$ is said to be a solution to the Bellman-equation if it satisfies

$$v(x) = \min_{A(x)} \left[ c(x, a) + \beta \int_X v(y) Q(dy|x, a) \right]$$

# Markov Control Processes

There are three cases of interest:

1. State space and action space are finite
2. Infinite state and actions, but cost-function is bounded and continuous
   Stokey and Lucas' textbook gives conditions to ensure existence of value function
3. Infinite state and actions, but cost-function not necessarily bounded or continuous

# Numerical dynamic programming

- As for theoretical results, it is also crucial here whether we assume finite or infinite state/action space
- Also important whether functions can be expected to be smooth/concave etc
- We consider 2 cases: Finite states and actions (FDP) and smooth and concave problems (EDP) (E for easy)

# FDP: Value function iteration

- State space is $X = \{1, \ldots, S\}$, action sets are $A(x)$ finite sets, $c(x, a)$ finite for all $x, a \in A(x)$. Transition probabilities are $\pi(x'|x, a)$. $\beta < 1$
- For any $v \in \mathbb{R}^S$ define $Tv \in \mathbb{R}^S$ by

$$Tv(x) = \min_{a \in A(x)} c(x, a) + \beta \sum_{x'} \pi(x'|x, a) v(x')$$

- Given and $v_0 \in \mathbb{R}^S$ value function iteration computes

$$v^* = \lim_{n \to \infty} T^n v_0$$

- The limit exists, is unique and satisfies $v^*(x) = Tv^*(x)$ for all $x$.

# FDP: Value function iteration - comments

- Convergence of value function iteration follows from the contraction mapping theorem.
- This is also true in function spaces, but here it is obvious, since $\beta < 1$

# FDP: Policy iteration

- Same setup, define policy $\alpha \in \mathbb{R}^S$, $\alpha(x) \in A(x)$. Define

$$v_\alpha(x) = E_x^\alpha \sum_{t=0}^{\infty} \beta^t c(x_t, \alpha(x_t))$$

  and $T_\alpha v(x) = c(x, \alpha(x)) + \beta \sum_{x'} \pi(x'|x, \alpha(x)) v(x')$
- Start with arbitrary (feasible) $\alpha_0 \in \mathbb{R}^S$ and choose $\alpha_{k+1}$ so that

$$T_{\alpha_{k+1}} v_{\alpha_k} = T v_{\alpha_k}$$

  Define $\alpha^*$ to satisfy $v_{\alpha^*} = v^*$

# FDP: Policy iteration – Theorem

For any $\alpha_0$ there exists a $N_0$ such that for all $k > N_0$

$$\alpha_k = \alpha^*$$

- We must have $T_{\alpha_{k+1}} v_{\alpha_k} \leq v_{\alpha_k}$
- If $T_{\alpha_{k+1}} v_{\alpha_k} = v_{\alpha_k}$ then $v_{\alpha_k} = v^*$. Otherwise we have $T_{\alpha_{k+1}} v_{\alpha_k} < v_{\alpha_k}$ and at least one improvement can be made
- Since action space if finite, must terminate in finitely many steps

# FDP: Real time DP

The following converges to the true value function with probability one. Fix $v_0, x_0$.

- Given any $v_t, x_t$, compute

$$a_t = \arg\min c(x_t, a) + \beta \sum_{x'} \pi(x'|x_t, a) v_t(x')$$

- Update $v_t$

$$v_{t+1}(x) = \begin{cases} c(x_t, a_t) + \beta \sum_{x'} \pi(x'|x_t, a_t) v_t(x') & \text{if } x = x_t \\ v_t(x) & \text{otherwise.} \end{cases}$$

- Sample $x_{t+1}$ from $\pi(.|x_t, a_t)$
- More on Friday

# EDP

- Suppose the state can be written as $x = (s, y)$ and conditional on $s$ the future $y'$ is a deterministic function of $a$. We have

$$v^*(x) = \min_{a \in A(x)} c(x, a) + \beta \sum_{s'} \pi(s'|s) v^*(x') \text{ s.t. } y' = p(x, a)$$

- Action space $A(x) \subset \mathbb{R}^l$ convex for all $x$
- Admissible endogenous states $y \in Y \subset \mathbb{R}^k$, $s_t$ follows Markov chain
- Assume $c(.)$ is bounded and continuous

# EDP: Value Function Iteration – The Idea I

For 'well behaved problems' we have that

- The Bellman operator has a unique fixed point $v_0$ in the space of bounded continuous value functions $C(X)$
- Starting from any function in $C(X)$ and repeatedly applying $T$ brings us closer and closer to the fixed point
- Thus, we will finally converge (by any desired precision)

# The Value Function: The Idea II

Thus, we can take any $v_0 \in C(X)$ and repeatedly apply the Bellman operator $T$ until we reach the desired level of convergence.
$\Rightarrow$ Value function iteration is a robust and reliable method!

However, in theory, we would need 'perfect approximations' for the value function in each iteration. In practice, we will either:

- Discretize the state space, i.e. allow only discrete choices and no values in-between. What happens with very large, finite state-space?
- Interpolate the value function, i.e. the update will only be 'perfect' at the interpolation nodes.

# The Stochastic Ramsey Model reconsidered

$$\max \mathbb{E}_0[U(\{c_t\})] \quad \text{s.t. } c_t + k_{t+1} \leq \underbrace{z(s_t)f(k_t) + (1-\delta(s_t))k_t}_{\equiv \bar{f}(s_t,k_t)},$$

$$c_t \geq 0,\ k_{t+1} \geq 0\ \forall t \in \mathbb{N}_0$$

$$k_0, s_0 \text{ given}$$

where the expectation is over the sequence of stocks $\{s_t\}_{t=1}^{\infty}$ given $s_0$.

Assume for now that $s_t$ follows a Markov chain with finite support. The only continuous state variable is capital. We have $S$ value functions that are functions of capital alone.

# Continuous versus discrete shocks

- For now we assume that the true stochastic process has finite support
- In applied work one typically assumes AR(1) or other continuous processes
- Our assumption can be viewed as a discretization of continuous shock (Tauchen and Hussey), but there are problems with this
- Will also look at continuous shocks but there we also need to approximate integrals $\rightarrow$ numerical integration

# Value Function Iteration with Discrete Spate Space

0. Choose:
   a) Capital grid $k = \{k_1, k_2, \ldots, k_n\},\ k_j < k_{j+1}$
   b) Guess for value functions $V_s^{(0)} = \{V_s^{(0)}(k_1), V_s^{(0)}(k_2), \ldots, V_s^{(0)}(k_n)\}$, $s = 1, ..., S$
   c) Stopping rule parameter $\epsilon > 0$
1. Update the value function
   1. $\forall s, i, j$: if $\bar{f}({}_s k_i) - k_j > 0$, compute

$$T_{i,j} = u(\bar{f}_s(k_i) - k_j) + \beta \sum_{s'} \pi(s'|s) V_{s'}^{(0)}(k_j),$$

      else $T_{i,j} = -huge$
   2. $\forall i$: $V_s^{(1)}(k_i) = \max\{\{T_{i,j}\}_{j=1}^n\}$; store index of maximum, $j_i^{max}$, as identifier of optimal choice, $k_{j_i^{max}}$
2. Check stopping rule: if $\| V^{(1)} - V^{(0)} \|_\infty < \epsilon(1-\beta)$, stop and report results; else $V^{(0)} = V^{(1)}$ and go to step 1.

# Value Function Iteration with Interpolation

- Approximate value function with a polynomial
- Pick some small number of points and compute exact Bellman operator at these points
- Interpolate...

# Function approximation

- To solve continuous DP, it is important to have a good approximation of the value function
- Need to learn some basic facts about the approximation of continuous or smooth functions
- Will focus on the one-dimensional case today and discuss higher dimensions later

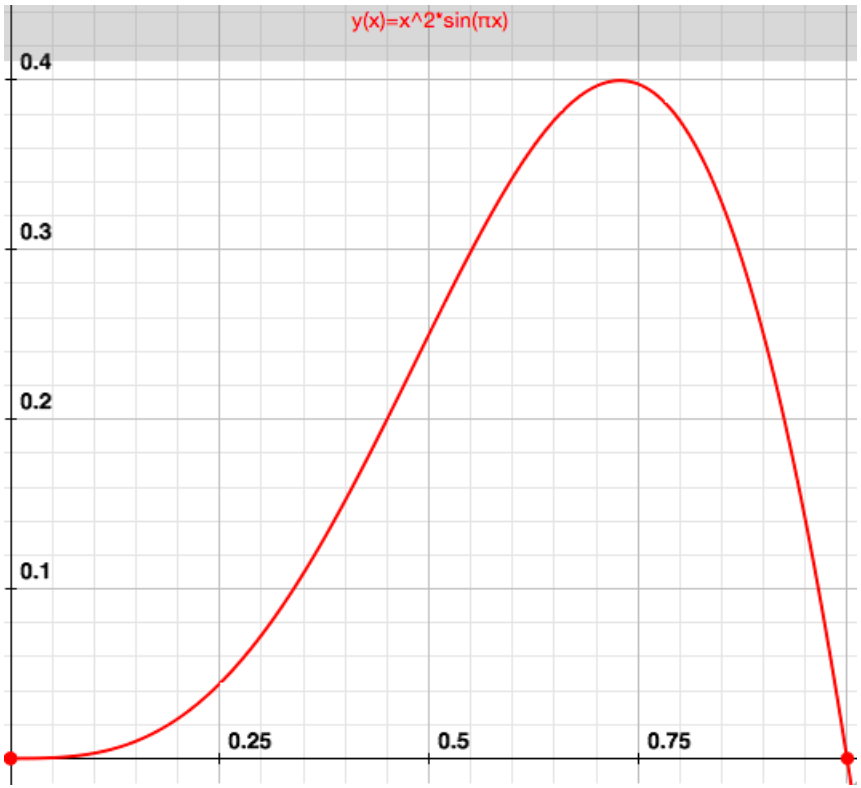
# Numerical Integration

- In economic problems with uncertainty, we typically assume that agents solve complicated integrals
- Computational economists need learn methods for numerical integration:
  - Monte Carlo
  - Quadrature and Cubature

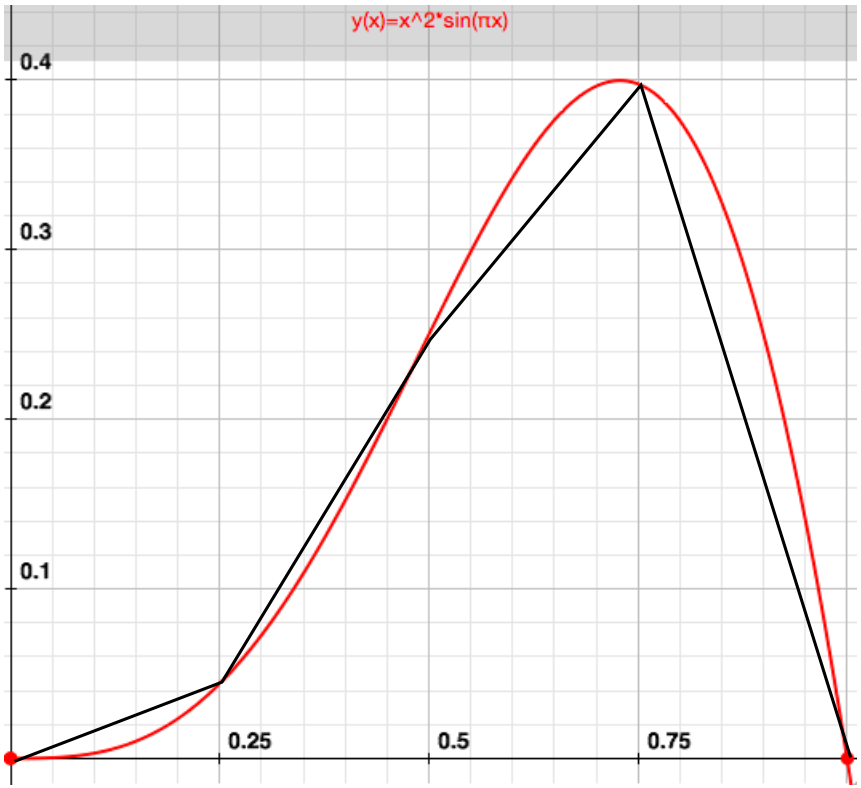
# Function approximation

- As it turns out one way to integrate complicated functions is to approximate them by simple functions whose integral we know!
- Later this week, we learn about other reasons why we want to approximate functions: Dynamic programming, projection methods etc...
- Will mostly focus on the one-dimensional case today and discuss higher dimensional approximation later
- How did you approximate functions in kindergarten?

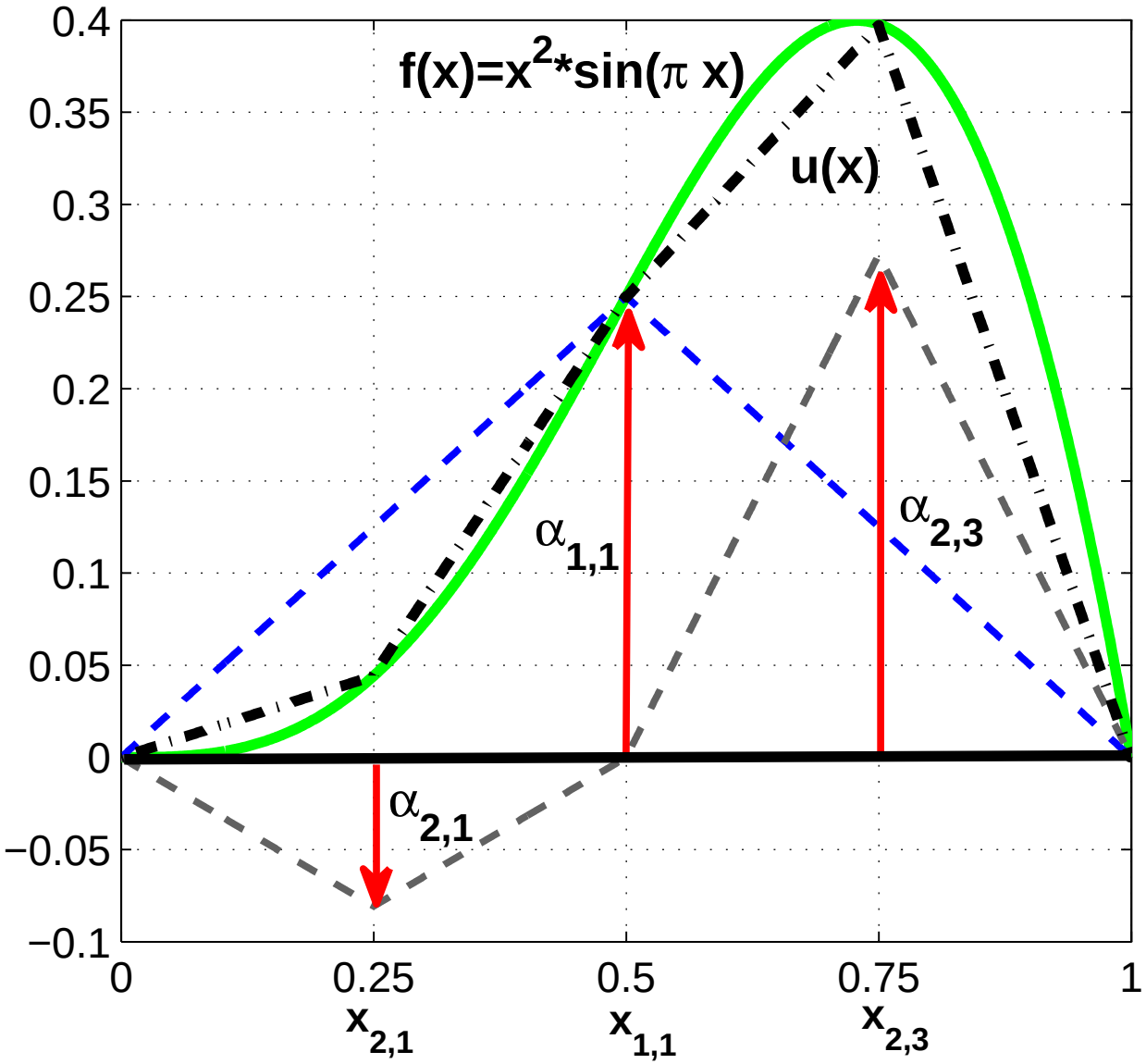
# Piecewise linear interpolation



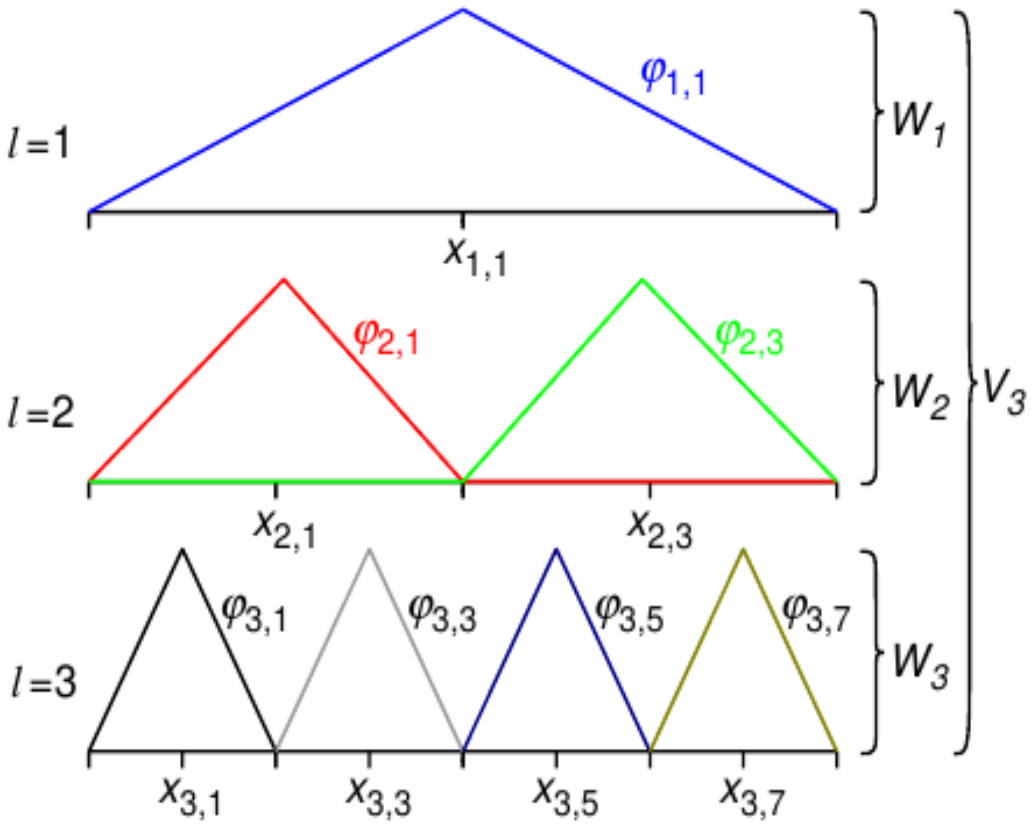
# Piecewise linear interpolation



# Piecewise linear interpolation



# Piecewise linear interpolation



# Piecewise linear interpolation

- In one dimension, we take as basic function on $[-1, 1]$

$$\phi(x) = \max(0, 1 - |x|)$$

and twist them to generate a family of basis functions on $[0, 1]$

$$\phi_{l,i}(x) = \phi(2^l x - i), i = 1, ..., 2^l - 1, \ i \text{ odd}$$

- Define

$$I_l = \{i \in \mathbb{N} : 1 \leq i \leq 2^l - 1, i \text{ odd}\}$$

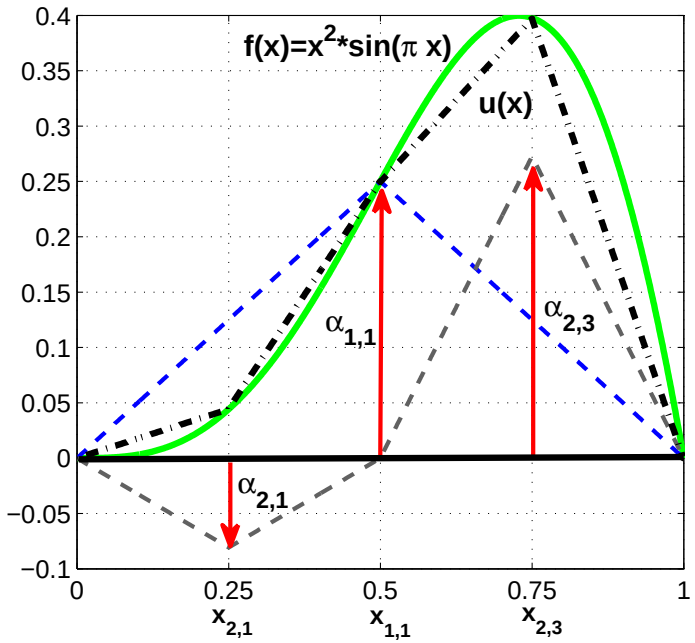
and

$$W_l = \text{span}\{\phi_{l,i}, i \in I_l\}$$

The space of piecewise linear functions is then

$$V_n = \bigoplus_{l \leq n} W_l$$

# Piecewise linear interpolation - coefficients




- $f(x) \simeq u(x) = \sum_{k=1}^{l} \sum_{i \in l_k} \alpha_{k,i} \phi_{k,i}(x)$
- The coefficients, $\alpha_{k,i}$ are *hierarchical surpluses*. They correct the interplant of level $l-1$ at $x_{l,i}$ to the actual value of $f(x_{l,i})$
- Become small as approximation becomes better

# Piecewise linear approximation

- Given the basis functions $\phi_{k,i}(x)$, we chose the $\alpha_{k,i}$ so that the approximating function matches the true function at a predetermined set of points
- Beyond Kindergarten, one can think of other good approximations, e.g. some norm on function space
- Which norm to take? We'll briefly talk about 2-norm, 1-norm and sup-norm.

# Uniform approximation

- Piecewise linear approximation seems a bit odd if the true function is sufficiently smooth.
- Even if the function is non-linear, one can approximate it arbitrarily well by polynomials. Weierstrass Theorem: Given any continuous function $f : [a, b] \to \mathbb{R}$ and any $\epsilon > 0$ there exists a polynomial $p(x)$ such that

$$\max_{x \in [a,b]} |f(x) - p(x)| < \epsilon$$

- How do we find this polynomial?
- In general that is tough, but can do it on a finite set of points - then one can also consider least squares, or least first power approximation.

# Polynomial interpolation 1

- We want to approximate a smooth function $f : [-1, 1] \to \mathbb{R}$ by a polynomial of degree $d$.
- In order to do so, we will interpolate $n$ of its function values, i.e. given some points $x_i, f(x_i)$, we try to find a polynomial that matches the function values at the points $x_i$.
- Polynomial is uniquely pinned down by $d + 1$ distinct points
- Alternatively we could take more than $d + 1$ points and do least square – for now we want to consider interpolation

# Polynomial interpolation 2

- Interpolate $n$ points by a univariate polynomial of degree $n-1$,

$$p_{n-1}(x) = \sum_{j=0}^{n-1} \theta_j x^j$$

- To find the unknown $n$ coefficients $(\theta_0, \ldots, \theta_{n-1})$, we could use the $n$ equations

$$y_i = \sum_{j=0}^{n-1} \theta_j (x_i)^j \text{ for } i = 1, \ldots, n$$

- Unfortunately, the 'Vandermonde' matrix

$$V = \begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} \end{pmatrix}$$

is typically extremely badly conditioned

# Polynomial interpolation 3

- Better to use Lagrange polynomials.
- Let the i'th Lagrange polynomial be defined by

$$l_i(x) = \prod_{j=1, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$

- Note that $l_i(x_j) = 1$ if and only if $i = j$ and it is zero otherwise.
- Therefore we can simply set

$$g(x) = \sum_{i=1}^{n} g(x_i) l_i(x).$$

- Simplest way to get an interpolation polynomial, but evaluation of $l_i$ could be costly..

# Orthogonal polynomials

- A good choice of polynomials are orthogonal under some inner product, in fact we have

$$\int_{-\infty}^{\infty} e^{-x^2} l_m(x) l_n(x) = 0 \text{ if } m \neq n$$

- Define Legendre polynomials as $P_0(x) = 1$, $P_1(x) = x$ and

$$P_{n+1}(x) = \frac{1}{n+1}\left((2n+1)xPn(x) - nP_{n-1}(x)\right)$$

We have

$$\int_{-1}^{1} P_m(x) P_n(x) dx = \frac{2}{2n+1}\delta_{mn}$$

- Define Chebychev terms as follows. Let $T_0(x) = 1$, $T_1(x) = x$, and recursively $T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x), \quad i = 1, ..., n$. We have

$$\int_{-1}^{1} (1-x^2)^{-1/2} T_m(x) T_n(x) dx = 0 \text{ if } m \neq n.$$

# Polynomial Interpolation 4

- Define Chebychev terms as follows. Let $T_0(x) = 1$, $T_1(x) = x$, and recursively

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

- Then set

$$g(x) = \sum_{i=0}^{n-1} \xi_i T_i(x)$$

with

$$\xi_i = \frac{2}{d_i(n-1)} \sum_{j=0}^{n-1} \frac{1}{d_j} T_i(x_j) g(x_j)$$

with $d_0 = f_{n-1} = 2$ and $d_i = 1$ otherwise.

# Polynomial interpolation 5

- Can write the approximating polynomials in many different, efficient ways.
- Much more important issue which polynomials to use is at which points to interpolate the function one wants to approximate.
- While it is true that one can approximate continuous functions arbitrarily well by polynomials, it is not true that one can do so by interpolating equi-spaced points.

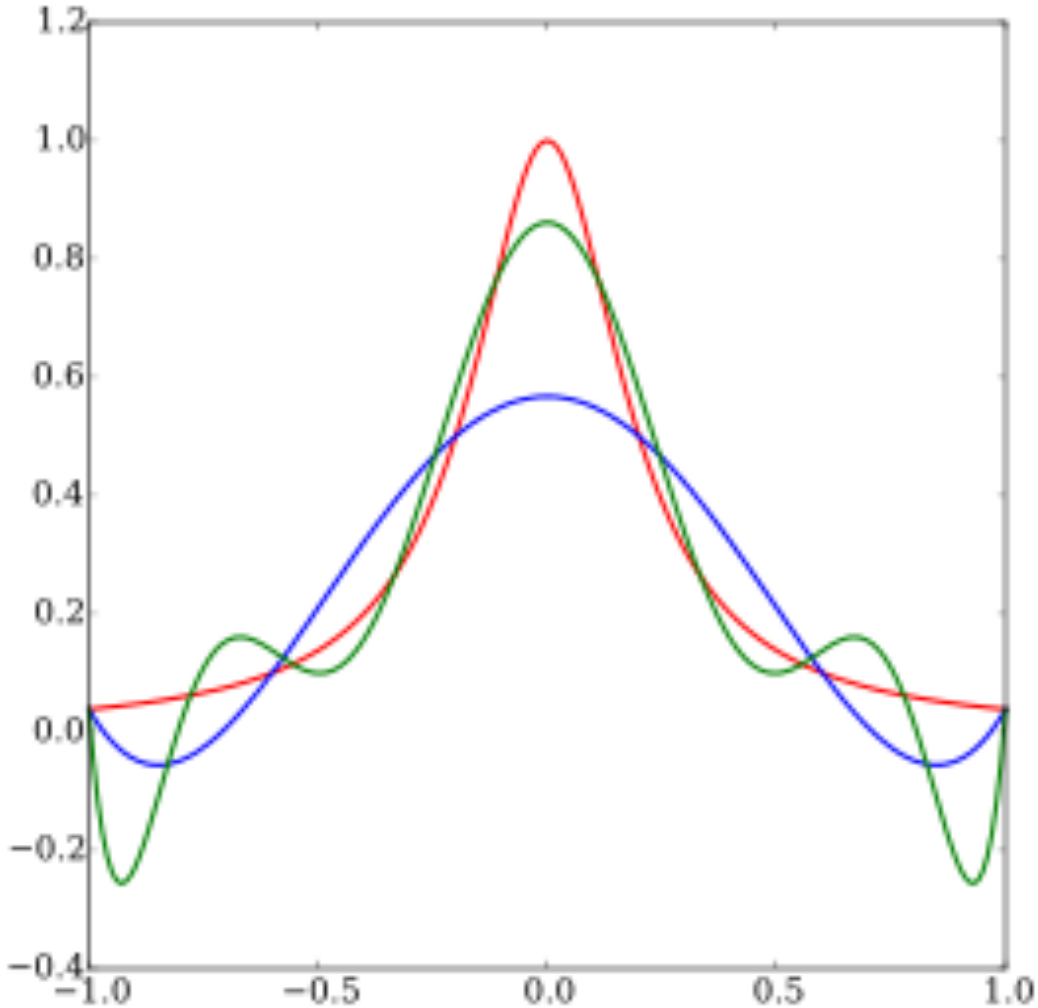
## Example (Runge)

Consider the function

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

on the interval $[-1, 1]$.

# Runge



# Polynomial interpolation 6

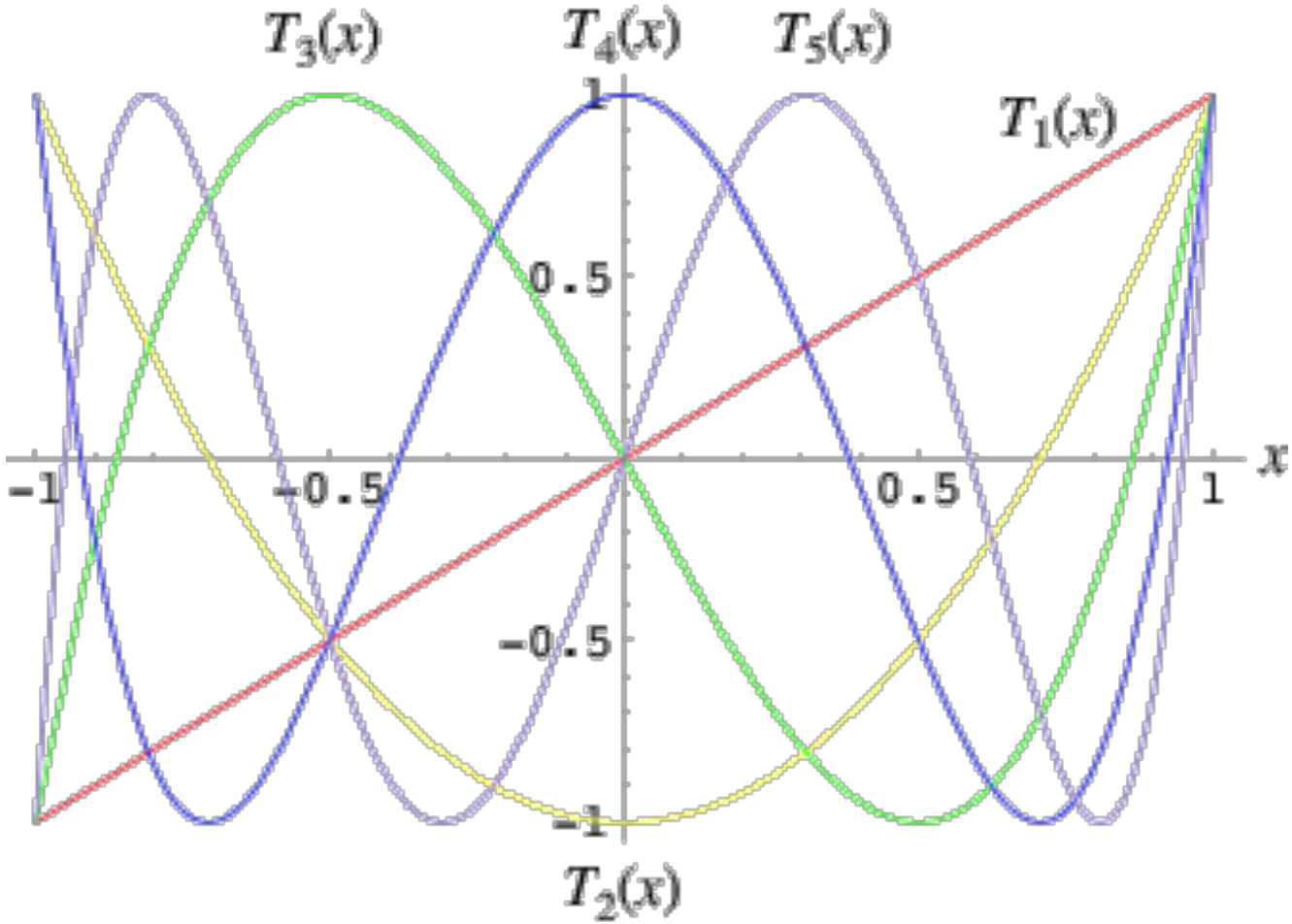
- It turns out that there are two good choices of points for which the problem does not occur.
- The first one is to use the zeros of Chebychev polynomials. These are given by

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

- An alternative, which is as good, is to use extrema of Chebychev polynomials. These are given by

$$z_k = -\cos\left(\frac{\pi(k-1)}{m-1}\right), \quad k = 1, ..., m, \quad m > 1.$$

# Chebychev points



# Some math to impress grandma

- Given a continuous function $f : [a, b] \to \mathbb{R}$ and $N$ points $z_1, \ldots, z_N \in [a, b]$, there is a unique interpolating polynomial of degree $N - 1$, $g(x)$. There is also the best uniform approximation of $f$ in the space of polynomials of degree $N$, let's call this $h(x)$.
- Under the sup-norm we have

$$\|f - g\| \leq (1 + \Lambda)\|f - h\|,$$

where $\Lambda$ is the Lebesgue constant

$$\Lambda = \max_{x \in [a,b]} \sum_{j=1}^{N+1} |l_j(x)|$$

- In principle, one can compute the optimal points by minimizing the Lebesgue constant. In practice it turns out that Chebychev points have a Lebesgue constant close to the optimal one.

# Limits to polynomial interpolation

- Consider the continuous function

$$f(x) = \sqrt{|x|}, x \in [-1, 1]$$

- Using Chebychev zeros, in order to approximate this function so that the sup-norm between approximation and function is less than $10^{-3}$ one needs 1.1 million points !!!!
- $\rightarrow$ Splines

# Splines 1

- A function $s : [a, b] \to \mathbb{R}$ is a spline of order $n$ if it is $C^{n-2}$ on $[a, b]$ and there are nodes $a = x_0 < x_1 < ..., x_n = b$ such that $s(x)$ is polynomial of degree $n - 1$ in each subinterval $[x_i, x_{i+1}]$
- Obviously, we can use splines to interpolate points $(x_i, y_i)_{i=0}^{n}$.
- The easiest scheme is piece-wise linear. This can be a good way to approximate difficult functions – will get back to this soon.

# Splines 2

- Want to focus on cubic splines (i.e. splines of order 4) that consist of cubic polynomials $p_1, ..., p_n$ such that

$$p_i(x_i) = y_i, \quad p_i(x_{i+1}) = y_{i+1}$$

and for all $i = 1, ..., n-1$

$$p_i'(x_i) = p_{i-1}'(x_i), p_i''(x_i) = p_{i-1}''(x_i)$$

- Since each cubic spline has 4 unknown coefficients, we have $4n$ unknowns and $2n + 2n - 2$ equations. Hm – too many unknowns.
- We need to impose some more conditions. Natural splines impose $s''(x_0) = s''(x_n) = 0$ to pin down the two remaining unknown coefficients We can also impose $s'(x_0) = y_0', s'(x_n) = y_n'$ – these are called Hermite splines.

# Splines 3: B-splines

- Can represent piecewise polynomial functions as the weighted sum of basis (B-) splines.
- Let $(t_j)$ be a non-decreasing (knot)-sequence. The j'th B-spline of order $k$ for $(t_j)$ is denoted by $B_{j,k,t}$ and they can be defined recursively by

$$B_{i,k}(x) = \frac{x - x_i}{x_{i+k} - x_i} B_{i,k-1}(x) + \frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} B_{i+1,k-1}(x)$$

with

$$B_{i,0}(x) = \begin{cases} 0, & x < x_i \\ 1, & x_i \leq x \leq x_{i-1} \\ 0, & x \geq x_i \end{cases}$$

- With this we can write

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i B_{i,k}(x)$$

where the coefficients $\alpha_i$ can be obtained from solving a linear system of equations.

# Higher dimensions..

- Can we use polynomials to approximate 10-dimensional functions? What about 100 dimensions?
- While interval is the clear domain in one dimension, in several dimensions could have different geometric structures.
- Typically one focuses on the cube $[-1, 1]^d$ but this obviously can create problems.
- Almost nothing is known about points with low Lebesgue constant in higher dimensions...

# Tensor products

- The simplest approach to multi-variate interpolation is to interpolate over a equi-spaced dense grid and to use a tensor product of one-dimensional polynomials.
- To approximate a function $f : [-1, 1]^d \to \mathbb{R}$ one can interpolate the function values at $n^d$ grid points by a polynomial of total degree$(n-1)^d$.
- The tensor product for $d$ dimensions is given by the set

$$P_n^d = \{p(x) | p(x) = \Pi_{i=1}^d p_n(x_i) \text{ for } p_n(x_i) \in P_n\}$$

# Not as hard as it looks

- Try out polynomial interpolation. Very simple with numpy: numpy.polynomial.chebyshev.Chebyshev.interpolate
- Try out scipy.interpolate.CubicSpline
- Once it is done, it looks simple and works much better than discretization!!!

# Back to the Ramsey problem

- We are now in a position to solve the Ramsey problem. Suppose we use polynomials of degree $d$ to approximate the value function
- Given some upper and lower bounds for capital $\underline{K} < \overline{K}$ define value function on $[\underline{K}, \overline{K}]$, translate Chebychev zeros to obtain $n = d + 1$ points $k_1, \ldots, k_n \in [\underline{K}, \overline{K}]$
- Fix some initial coefficients $\theta^0$.
- Iterate, given $V(., \theta^i)$ solve for $V^{i+1}$ at all points $k_1, ..., k_n$ and for all values of the shock $z \in \mathcal{Z}$. Interpolate to find $\theta^{i+1}$
- Hope that it converges (it should)

# Easy dynamic programming - comments

- Easy to extent the method to higher dimensions and/or continuous shocks
- Need good function approximation in high dimensions (e.g. Smolyak)
- Need good solvers for Bellman equation
- Need good domains!
- Problems can arise due to non-differentiabilities
- How important is shape-preservation? see e.g. Cai and Judd (2010) and references therein.

# An alternative way to solve the Ramsey problem

- Instead of doing slow value function iteration, can solve for the policy function.
- Can transform the problem into a functional equation
- $\rightarrow$ projection methods

# Reducing EDP to a functional equation

- Suppose the state can be written as $x = (s, y)$ and conditional on $s$ the future $y'$ is a deterministic function of $a$. We have

$$v^*(x) = \min_{a \in A(x)} c(x, a) + \beta \sum_{s'} \pi(s'|s) v^*(x') \text{ s.t. } y' = p(x, a)$$

- In 'good cases' can consider the first order condition

$$\frac{\partial c(x, a)}{\partial a} + \beta \sum_{s'} \pi(s'|s) \frac{\partial c(x', a')}{\partial y} \frac{\partial p(x, a)}{\partial a} = 0$$

(envelope theorem !)

- Write this as function equation in $\alpha(x)$

# An alternative way to solve the Ramsey problem

- Instead of doing slow value function iteration, can solve for the policy function.
- Can transform the problem into a functional equation
- $\rightarrow$ projection methods. Possibly quadratic convergence !

# The Deterministic Ramsey Model

Choose $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ to maximize $U(\{c_t\}_{t=0}^{\infty})$ subject to

$$\forall t \in \mathbb{N}_0 : 0 \leq k_{t+1} \leq \underbrace{f(k_t) + (1-\delta)k_t}_{\equiv \bar{f}(k_t)} - c_t, 0 \leq c_t, k_0 \text{ given}$$

where:

- $c_t$ is consumption at time $t$
- $U(\{c_t\}_{t=0}^{\infty})$ is utility of the consumption stream $\{c_t\}_{t=0}^{\infty}$
- $k_t$ is the capital stock at time $t$, and $k_0$ the initial capital stock
- $f(\cdot)$ is the production function
- $\bar{f}(\cdot)$ is production including non-depreciated capital
- $\delta$ is depreciation

# Standard Assumptions on Preferences and Production

## Production

- Neoclassical Production:
  $f(0) = 0$, $f \in C^2(\mathbb{R})$,
  $f'(k) > 0$, $f''(k) < 0$,
  $\lim_{k\to 0} f'(k) = \infty$,
  $\lim_{k\to\infty} f'(k) = 0$
- Special Case:

$$f(k) = k^{\alpha}$$

Cobb-Douglas with capital share $\alpha$ and fixed labor supply (normalized or intensive form)

## Preferences

- Time-separable utility:

$$U(\{t\}_{t=0}^{\infty}) = \sum_{t=0}^{\infty} \beta^t u(c_t)$$

with discount factor $0 < \beta < 1$, $u'(c) > 0$, $u''(c) < 0$, and $\lim_{c\to 0} u'(c) = \infty$.

- Special Case:

$$u(c_t) = \begin{cases} \ln(c_t), & \gamma = 1 \\ \frac{c_t^{1-\gamma}}{1-\gamma}, & \gamma \in \mathbb{R}_+ \setminus \{1\} \end{cases}$$

CRRA utility

# The Euler Equation

Due to the above assumptions:

- $c_t \geq 0,\ k_{t+1} \geq 0$ are never binding
- the budget constraint is always binding: $c_t = \bar{f}(k_t) - k_{t+1}$

Therefore, the **Lagrangian** of the maximization problem simplifies to:

$$\mathcal{L} = \sum_{t=0}^{\infty} \beta^t [u(c_t) + \lambda_t(\bar{f}(k_t) - c_t - k_{t+1})]$$

$$\frac{\partial \mathcal{L}}{\partial c_t} = 0 \Leftrightarrow u'(c_t) = \lambda_t;\ \frac{\partial \mathcal{L}}{\partial k_{t+1}} = 0 \Leftrightarrow \lambda_t = \beta \lambda_{t+1} \bar{f}'(k_{t+1})$$

Combining, we get the **Euler equation**(s):

$$u'(\bar{f}(k_t) - k_{t+1}) = \beta \bar{f}'(k_{t+1}) u'(\bar{f}(k_{t+1}) - k_{t+2}) \quad \forall t \in \mathbb{N}_0$$

# Recursive Equilibrium

Hard to solve for an infinite sequence directly!

$\Rightarrow$ Reduce problem to two periods: 'today' and 'tormorrow'

$\Rightarrow$ Suppose optimal choice does not depend on $t$ directly, just on $k_t$

$\Rightarrow$ Look for recursive equilibrium with capital $k$ as endogenous state

$\Rightarrow$ A recursive equilibrium consumption function $C(k)$ must satisfy:

$$u'(C(k)) = \beta \cdot \bar{f}'(\bar{f}(k) - C(k)) \cdot u'(C(\bar{f}(k) - C(k)))$$

# The Steady State

In a steady state, $k^*$, capital does not change from 'today' to 'tormorrow':

$$\bar{f}(k^*) - C(k^*) = k^*$$

This requirement and the Euler equation determine the steady state:

$$\begin{aligned} u'(C(k^*)) &= \beta \cdot \bar{f}'(k^*) \cdot u'(C(k^*)) \\ 1 &= \beta \cdot \bar{f}'(k^*) \\ k^* &= (\bar{f}')^{-1}(\tfrac{1}{\beta}) \end{aligned}$$

and therefore:

$$c^* = C(k^*) = \bar{f}(k^*) - k^*$$

# Local solution: Taylor's Theorem

## Taylor's Theorem

If $f \in C^{n+1}[a,b]$ and $x, x_0 \in [a,b]$, then

$$\begin{aligned} f(x) =& f(x_0) + (x - x_0) f'(x_0) + \frac{(x - x_0)^2}{2} f''(x_0) \\ &+ \ldots + \frac{(x - x_0)^n}{n!} f^{(n)}(x_0) + R_{n+1}(x), \end{aligned}$$

with $R_{n+1}(x) = \mathcal{O}\left(|x - x_0|^{n+1}\right)$.

- $R_{n+1}(x) = \mathcal{O}\left(|x - x_0|^{n+1}\right)$ means that $\lim_{x \to x_0} \frac{R_{n+1}(x)}{|x - x_0|^{n+1}} = 0$
- More precisely, the truncation error is given by

$$\text{with } R_{n+1}(x) = \frac{1}{n!} \int_{x_0}^{x} (x - t)^n f^{(n+1)}(t) dt = \frac{(x - x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\xi)$$

# Implications of Taylor's Theorem

Taylor's theorem says that:

- One can use derivative information at a single point $x_0$ to construct a polynomial approximation of a differentiable function.
- The approximation tends to be better ...
  - for $x$ close to $x_0$.
  - when many derivatives are used.

# The (almost) trivial case: Taylor's Theorem for $n = 1$

## Taylor's Theorem for n=1

If $f \in C^2[a, b]$ and $x, x_0 \in [a, b]$, then

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \mathcal{O}\left(|x - x_0|^2\right),$$

- We can use the tangent to approximate a twice differentiable function
- If we want to approximate $C(k)$ around the steady state $k^*$, we have to find $C'(k^*)$

# Linear Approximation around the steady state

Euler equation at the Steady State:

$$u'(C(k^*)) = \beta u'(C(\bar{f}(k^*) - C(k^*)) \cdot \bar{f}'(\bar{f}(k^*) - C(k^*))$$

Differentiate with respect to $k = k^*$ and drop all arguments:

$$u'' \cdot C' = \beta u'' \cdot C'(\bar{f}' - C')\bar{f}' + \beta u' \bar{f}''(\bar{f}' - C')$$

Use that $\bar{f}' = \frac{1}{\beta}$ at the steady state and devide by $u''$:

$$0 = \underbrace{1}_{a} \cdot (C')^2 + \underbrace{(1 - \bar{f} + \beta \frac{u'}{u''} \bar{f}'')}_{b} C' - \underbrace{\frac{u'}{u''} \bar{f}''}_{c}$$

$$\Rightarrow C'(k^*) = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

From this we get the approximations:

$$\hat{C}(k) = c^* + (k - k^*) \cdot C'(k^*)$$

$$\text{OR: } \hat{K}^+(k) = k^* + (k - k^*) \cdot (1 - C'(k^*))$$

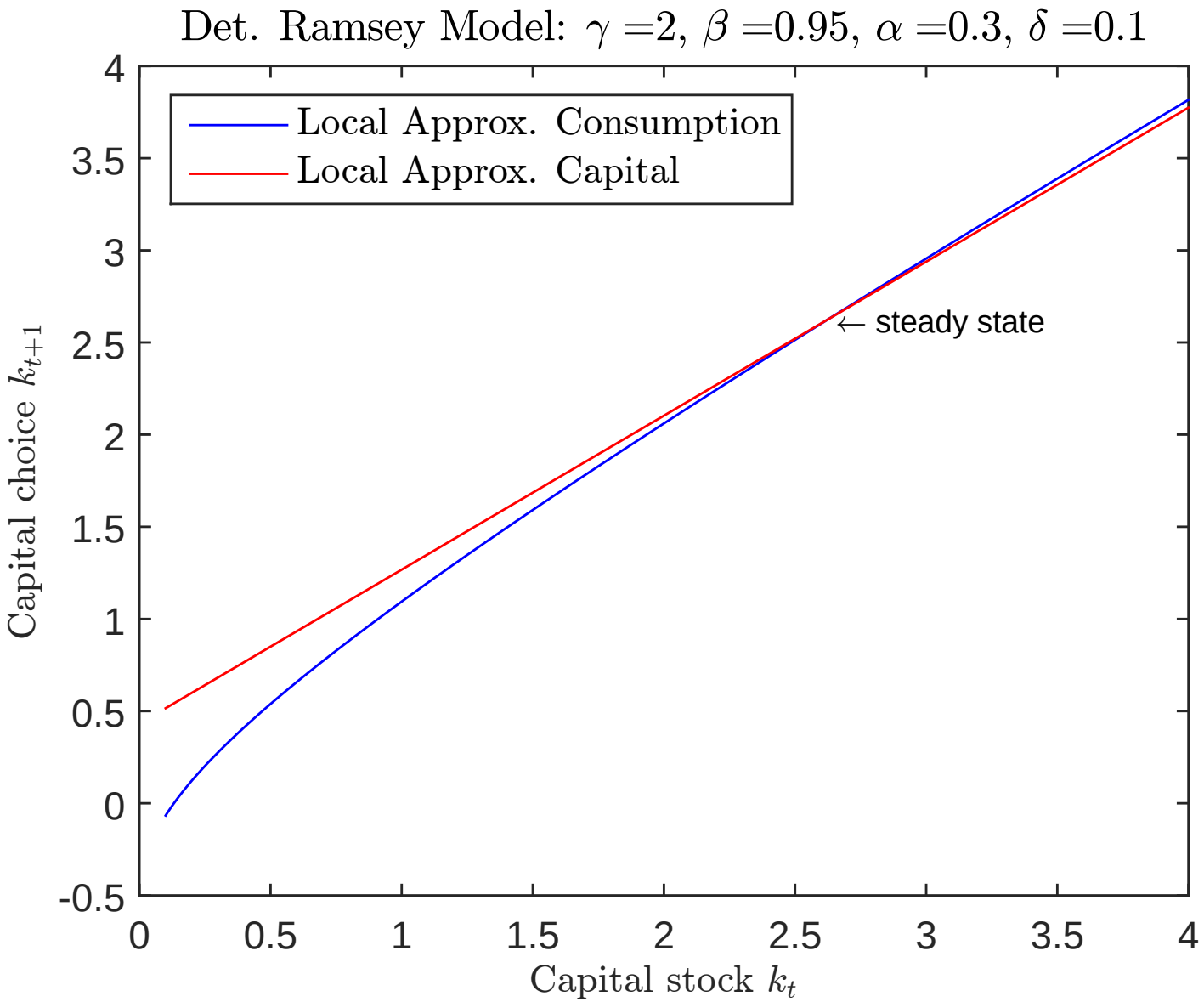
# Higher Order Approximations around the steady state

- Take higher-order derivatives of the Euler equation at the Steady State
- Get higher-order terms of the Taylor expansion:

$$\hat{C}(k) = c^* + (k - k^*) \cdot C'(k^*) + \frac{(k - k^*)^2}{2} \cdot C''(k^*) + \dots$$

See Judd's textbook, special issue of JEDC

# Linear Approximations around the Steady State



# Assessing Accuracy: Euler Errors I

We want a policy function $C(\cdot)$ that satisfies the Euler equation

$$u'(C(k)) = \beta \cdot \bar{f}'(\bar{f}(k) - C(k)) \cdot u'(C(\bar{f}(k) - C(k)))$$

at all $k \in [k_{min}, k_{max}]$, not only at $k^*$. We proceed as follows:

- Create many points $\{\tilde{k}_i\}_{i=1}^{I} : \tilde{k}_i \in [k_{min}, k_{max}]$
- Compute consumption implied by approximate policy: $\hat{c}_i = \hat{C}(\tilde{k}_i)$.
- Compute consumption implied by Euler equation and approximate policy 'tomorrow': $c_i^* = u_c^{-1}\left[\beta \bar{f}'(\bar{f}(\tilde{k}_i) - \hat{c}_i) \cdot u_c\left(\hat{C}(\bar{f}(\tilde{k}_i) - \hat{c}_i)\right)\right]$
- The (relative) error that the agent makes 'today' given his choice 'tomorrow' is the Euler error:

$$E_i = |\frac{\hat{c}_i}{c_i^*} - 1|$$

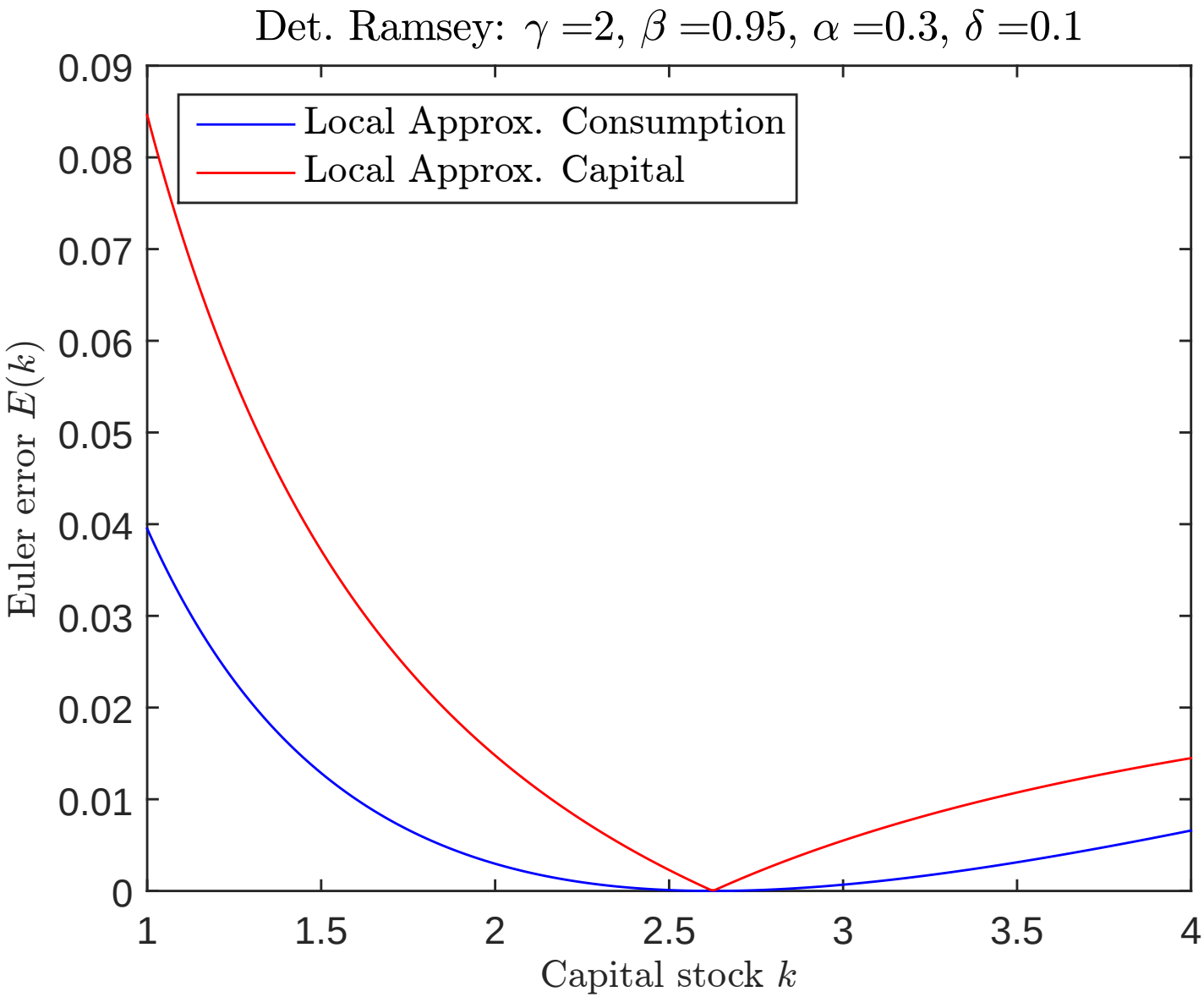
# Assessing Accuracy: Euler Errors II

- Choose points $\{\tilde{k}_i\}_{i=1}^{I}$ either
  - randomly (uniformly distributed) in $[k_{min}, k_{max}]$, or
  - as a very fine (equidistant) grid on $[k_{min}, k_{max}]$
  - or along a simulated path
- 'Bounded rationality' interpretation: The Euler error

$$E_i = |\frac{\hat{c}_i}{c_i^*} - 1|$$

is the fraction by which the approximate consumption choice today differs from the optimal one (given the approximate consumption choice tomorrow). For instance, $E_i = 0.05$ means that consumption is 5% too high or too low relative to the optimum

# Euler Errors for Local Approximations



# Our $1^{st}$ Global Solution: Piecewise Linear Collocation

- The accuracy of the local solution is high close to the steady state, yet low further away
- We would like to force the solution to be accurate also further away from the steady state
- We demand that the solution satisfies the Euler equation exactly on a grid of points (instead of only at the steady state)
- As a start, we interpolate linearly between these points

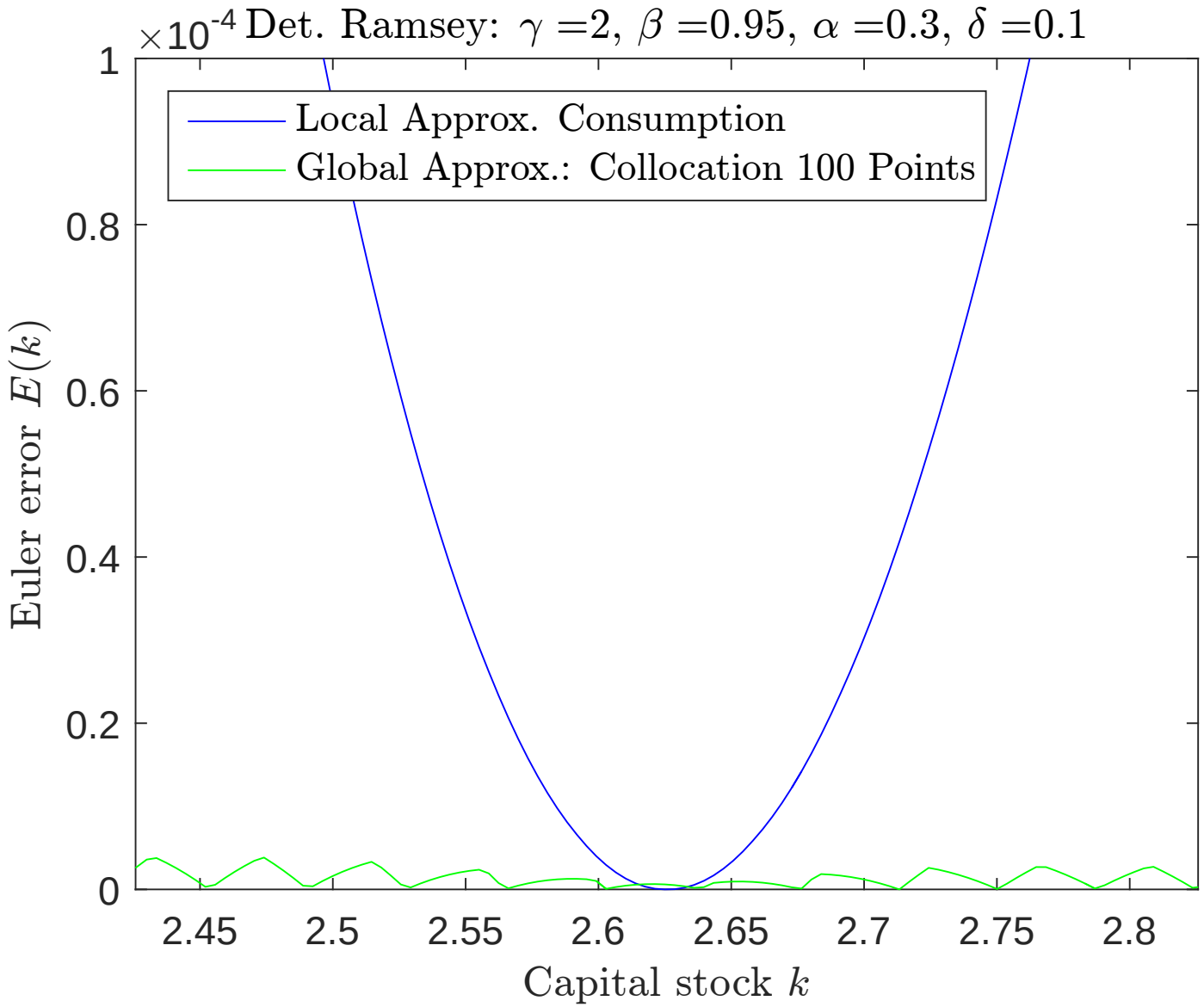
# Algorithm for Collocation with Piecewise Linear Basis

0. Initial Step *(Set grid, initial policy, and error tolerance)*
   a) Set capital grid $K = [K_1 \; K_2 \; \ldots \; K_n] \in \mathbb{R}^n_+$, $K_j < K_{j+1} \; \forall \; j$ and set guess for policy at gridpoints $K^+ = [K_1^+ \; K_2^+ \; \ldots \; K_n^+] \in \mathbb{R}^n_+$
   b) Set error tolerance $\bar{\epsilon} > 0$
1. Main Step *(Solve for parameters of policy function)*
   **Solve** for $K^+$ the system of non-linear equations, $\forall 1 \leq j \leq n$ :

$$R(K_j) \equiv u'(\bar{f}(K_j) - K_j^+) - \beta \cdot \bar{f}'(K_j^+) \cdot u'\left(\bar{f}(K_j^+) - PL(K_j^+; K, K^+)\right) = 0,$$

   where $PL(x; K, K^+) \equiv \frac{\left(K_{j+1}^+ - x\right)K_j^+ + \left(x - K_j^+\right)K_{j+1}^+}{K_{j+1}^+ - K_j^+}$ for $K_j \leq x \leq K_{j+1}$
2. Final Step *(Check error criterion)*
   a) Calculate error: $\epsilon = \max_j R_j$
   b) If $\epsilon < \bar{\epsilon}$, then stop and report result $(K, K^+)$; otherwise go to step 0 and make different choices.

# Comparing Global and Local Solution: Euler Errors



# Can We Do Better?—Chebyshev Polynomials

- The policy function we are interpolating seems very smooth, thus polynomials should do much better than linear splines
- Choose basis of orthogonal polynomials $\rightarrow$ Chebyshev polynomials

Recall:

- Chebyshev polynomials: $T_n(x) \equiv \cos\left(n \arccos(x)\right)$, for $n \geq 0$
- Chebyshev zeros of $T_n$: $Z_j^n \equiv -cos\left(\frac{2j-1}{2n}\pi\right)$, for $j \geq 1$:
- The degree $n-1$ Chebyshev interpolant of a function g (on $[-1,1]$):

$$CP(x;a) \equiv \sum_{l=0}^{n-1} a_l T_l(x), \text{where } a_l \equiv \frac{\sum_{j=1}^{n} g(Z_j^n) T_l(Z_j^n)}{\sum_{j=1}^{n} T_l(Z_j^n)^2}$$

# Algorithm for Collocation with Chebyshev Polynomials

0. Initial Step *(Set grid, initial policy, and error tolerance)*
   a) Set Chebyshev capital grid $K = [K_1 \; K_2 \; \ldots \; K_n] \in \mathbb{R}^n_+$ by choosing $n, \bar{K}, \underline{K}$ and setting $K_j = \left(Z_j^n + 1\right)\left(\bar{K} - \underline{K}\right)/2 + \underline{K}$ and set guess for parameters of policy function $a \in \mathbb{R}^n$
   b) Set error tolerance $\bar{\epsilon} > 0$
1. Main Step *(Solve for parameters of policy function)*
   **Solve** for $a$ the system of non-linear equations, $\forall 1 \leq j \leq n$ :

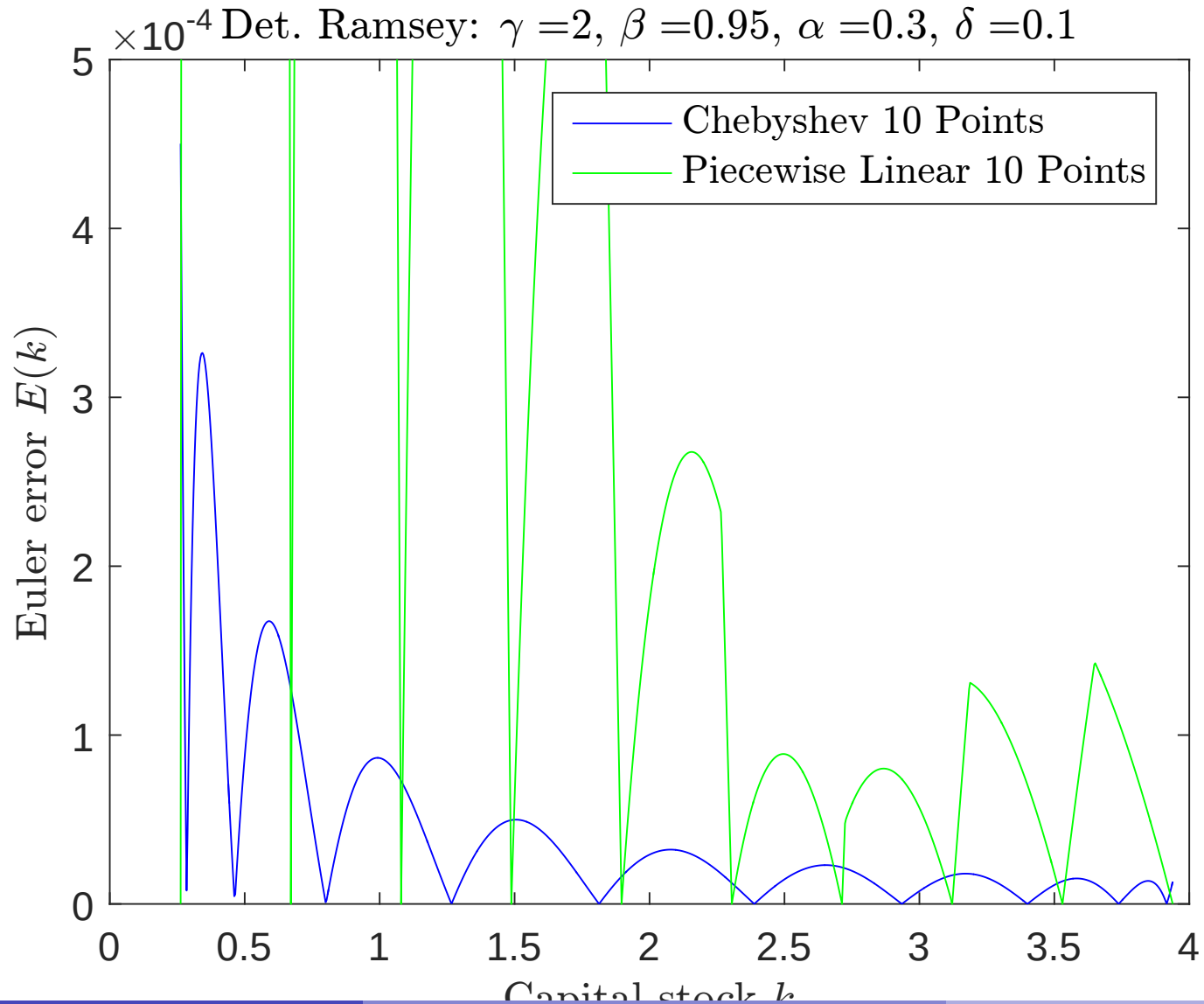
$$R(K_j; a) \equiv u'(\bar{f}(K_j) - K_j^+) - \beta \cdot \bar{f}'(K_j^+) \cdot u'\left(\bar{f}(K_j^+) - K_j^{++}\right) = 0,$$

   where

$$K_j^+ = CP\left(\rho(K_j); a\right), K_j^{++} = CP\left(\rho(K_j^+); a\right), \rho(x) = \frac{2(x - \underline{K})}{\bar{K} - \underline{K}} - 1$$

2. Final Step *(Check error criterion)*
   a) Calculate error: $\epsilon = \max_j R_j$
   b) If $\epsilon < \bar{\epsilon}$, then stop and report $a$; otherwise go to step 0.

# Comparing Piecewise Linear and Chebyshev Interpolation



# Collocation—A Special Case of Projection Methods

- As above, let $R(x; a)$ denote the Euler Equation residual at $x$ for the Chebyshev polynomial with coefficients $a$
- Then the Chebyshev collocation method solves for $a$ such that

$$\forall 1 \leq j \leq n : R(K_j; a) = 0$$

- Using the Dirac delta "function" this can be written as:

$$\forall 1 \leq j \leq n : P_j^C(a) \equiv \int_{\underline{K}}^{\bar{K}} R(x; a) \cdot \delta(x - K_j) dx = 0$$

- In general, we weight the residual with functions $\{w_j\}_{j=1...n}$ and aim for all these "projections" to become zero:

$$\forall 1 \leq j \leq n : P_j(a) \equiv \int_{\underline{K}}^{\bar{K}} R(x; a) w_j(x) dx = 0$$

# Time iteration collocation

- How do we solve for the coefficients????
- In this simple example can use Newton's method
- In more complicated examples this becomes tricky. System is large and ill-conditioned
- Back to iteration!!!

# A solution method - time iteration collocation

0: Select a d-box $\Theta$, a family of functions $\hat{F}$ which can be parametrized by numbers

1: Select a finite grid $\mathcal{G} \subset \Theta$ of collocation points and the parameters $\xi(0)$ for a starting $\hat{F}^0$

2: Given parameters $\xi(n)$ and thus the function $\hat{F}^n$, $\forall \theta \in \mathcal{G},\ \forall z \in \mathbf{Z}$, solve system

$$g\left(\theta, z, x, E_z\left[h\left(x, \hat{F}^n(x, z'), z'\right)\right]\right) = 0$$

for the unknown $x$

3: Compute the new coefficients $\xi(n+1)$ by interpolation of the solutions in 2

4: Check some stopping criterion, if not satisfied, go to 2

5: Conduct error analysis

# What is needed

- Approximation of policy functions
  - How to select $\Theta$: What variable to use? Domain?
  - How to select $\mathcal{G}$
  - Interpolation, picking $\xi(0)$
- Solving nonlinear equations

# Advantages of time iteration

- For very large systems, Newton-methods might not cut it
- Can adaptively refine the grid.
  Will show two examples, the most famous example (that I do not show) is probably:
  Carroll (2006EL)): Endogenous grid method: Use a discrete grid for the optimal choice and trace back the state that induces this choice ) get an "endogenous" grid for the state from exogenous grid on the choice

# Motivation for fancy endogenous grids

A global solution method for dynamic models with

- occasionally binding constraints,
- several continuous state variables.

$\Rightarrow$ Computational challenge: Kinks in policy functions

Johannes Brumm and Michael Grill (2014): Computing Equilibria in Dynamic Models with Occasionally Binding Constraints, JEDC 38, 142-160:

Main components of the algorithm:

- Adaptive grid scheme to add interpolation nodes at kinks
- Flexible interpolation technique based on Simplicial Interpolation

$\Rightarrow$ Called Adaptive Simplicial Interpolation (ASI).

# Short sale constraints

Suppose the Kuhn-Tucker conditions are necessary and sufficient. They are

$$D_x f(x) + \sum_j \lambda_j D_x g_j(x) + \sum_j \mu_j D_x h(x) = 0$$

$$\lambda_j \geq 0, g_j(x) \geq 0, \quad j = 1, ..., r$$

$$\lambda_j g_j(x) = 0, \quad , j = 1, ..., r$$

$$h_j(x) = 0, \quad j = 1, ..., s$$

# Garcia-Zangwill trick

Define

$$\alpha^+ = (\max[0, \alpha])^k, \quad \alpha^- = (\max[0, -\alpha])^k.$$

Note that $\alpha^+ \geq 0$, $\alpha^- \geq 0$ and $\alpha^+\alpha^- = 0$. Recast the Kuhn-Tucker conditions as follows.

$$D_x f(x) + \sum_j \alpha_j^+ D_x g_j(x) + \sum_j \mu_j D_x h(x) = 0$$

$$\alpha_j^- - g_j(x) = 0, \quad , j = 1, ..., r$$

$$h_j(x) = 0, \quad j = 1, ..., s$$

# A problem with forward looking constraints

$$\max_{(a_t)_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t r(x_t, a_t, s_t) \text{ subject to}$$

$$x_{t+1} = \zeta(x_t, a_t, s_t) \tag{1}$$

$$p(x_t, a_t, s_t) \geq 0 \tag{2}$$

$$g(x_t, a_t, s_t) + E_t \sum_{n=1}^{\infty} \beta^n g(x_{t+n}, a_{t+n}, s_{t+n}) \geq \underline{G}, \ \forall t \tag{3}$$

$$x_0 \text{ given .} \tag{4}$$

Assume that $(s_t)$ follows a finite Markov chain.

# Economic applications

- There is a single principal and an agent who contract at $t = 0$. The principal has within period payoff $r(x_t, a_t, s_t)$ which depends upon the state $x_t$, the action $a_t$ and the shock $s_t$. The agent's period payoff is $g(x_t, a_t, s_t)$. An incentive constraint has to hold every period.
- Agents trade in financial markets but trades must be incentive compatible in the sense that at each node agents walk away from their promise if autarky utility is higher than future expected utility from staying in the market (Kehoe-Levine Alvarez-Jermann). How can we determine (constrained) efficient allocations?

# Recursive solutions

- The standard approach to making recursive contracting problems (or game theoretic models) is to use continuation utility as a state variable.
- Marcet and Marimon (1994) develop an alternative recursive saddle point method which is recursive in a cumulation of Lagrangian multipliers.
- Equivalence turns out to rely on concavity assumptions about the original problem which insure strictly concave Pareto frontier.
- Method can be extended to weakly concave Pareto frontiers using lotteries (Cole and Kubler (2012)).

# Recursive in utility

The original problem is not recursive in the state $x_t$ because of the forward-looking constraint. It can be recursive in the ex ante promised utility to the agent $G$ (before $s$ is realized) and the state $x$.

$$V(G, x, s_-) = \max_{a \in A, (G'_s) \in ?} r(x, a, s) + \beta \sum_{s' \in \mathcal{S}} \pi(s'|s) V(G'_s, x', s')$$

subject to

$$g(x, a, s) + \beta \sum_{s \in \mathcal{S}} \pi(s'|s) G'_s \geq G$$

$$g(x, a, s) + \beta \sum_{s \in \mathcal{S}} \pi(s'|s) G'_s \geq \underline{G}$$

$$x' = \zeta(x, a, s)$$

$$p(x, a, s) \geq 0.$$

# Recursive in utility - comments

- Need to find some good space for the $G$...
- This can be computationally very costly
- Simpler to use mutlipliers/Negishi-weigths as state
  Pioneered by Marcet and Marimon, but they got a little confused about duality...

# The Lagrange dual function

Consider a finite dimensional constrained optimization problem

$$p^* = \min_f(x) \text{ s.t. } g(x) \leq 0,$$

with $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$.
Can define the Lagrangian

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

and dual function

$$h(\lambda) = \min_x L(x, \lambda)$$

Easy to see that for any $\lambda \geq 0$, $h(\lambda) \leq p^*$

# The dual problem

Consider now

$$d^* = \max_{\lambda \geq 0} h(\lambda)$$

Note that this is a convex problem and that by the above $d^* \leq p^*$.
We have that 'strong duality holds' if $f$ and $g$ are convex functions and if Slater's condition holds, i.e. there is some $x$ with $g(x) \ll 0$.

# A max-min interpretation

Note that

$$\sup_{\lambda \geq 0} L(x, \lambda) = \begin{cases} f(x) & \text{if } g(x) \leq 0 \\ \infty & \text{otherwise} \end{cases}$$

This obviously implies that

$$p^* = \inf_x \sup_\lambda L(x, \lambda)$$

So we have seen

$$\inf_x \sup_\lambda L(x, \lambda) \geq \sup_\lambda \inf_x L(x, \lambda)$$

and $=$ if there is no duality gap

# Saddle point

A saddle point of a function $f(w, c)$ is $w^*, c^*$ such that

$$f(w^*, c) \leq f(w^*, c^*) \leq f(w, c^*) \text{ for all } w, c$$

If $f(.)$ has a saddle point there is no duality gap and

$$\inf_w \sup_c f(w, c) = \sup_c \inf_w f(w, c)$$

So: if $(x, \lambda)$ is a saddle point of the Lagrangian then there is no duality gap and $(x, \lambda)$ are optimal solutions.

# Assumptions

1. The reward functions $r_s : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ and the constraint functions $g_s : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ are continuous and bounded (both from below and above) for all shocks $s \in \mathcal{S}$.
2. There is discounting, i.e. $\beta \in (0, 1)$.
3. There is an $\epsilon > 0$ such that for each initial condition $(x_0) \in \mathcal{X}$ there exists an action $(\psi_t)$ such that for all agents $i = 1, ..., I$,

$$\inf_{h^{t-1}, s_t, a_t \in \text{supp}(\psi_t)} \left[ g^i(x_t, a, s_t) + E_t^{\psi} \sum_{n=1}^{\infty} \beta^n g^i(x_{t+n}, a_{t+n}, s_{t+n}) - \bar{g}_1^i \right] > \epsilon$$

and

$$E_{-1}^{\psi} \sum_{t=0}^{\infty} \beta^t g^i(x_t, a_t, s_t) - \bar{g}_2^i > \epsilon.$$

# Lagrangian of forward looking problem

The Lagrangian for the dynamic problem (just using IC-constraints) is

$$L((\lambda_t),(a_t);x_0,s_0) =$$
$$E_0^{\psi}\sum_{t=0}^{\infty}\beta^t\left(r(x_t,a_t,s_t)+\lambda_t(\sum_{n=0}^{\infty}\beta^n g(x_{t+n},a_{t+n},s_{t+n})-\underline{G})\right)$$

# Saddle point for forward looking problem

- If $[(\lambda_t)^*, (a_t)^*]$ is a saddlepoint, then

$$L\left[(\lambda_t)^*, (a_t)^*\right] = \min_{(\lambda_t)} \max_{(a_t)} L\left((\lambda_t), (a_t)\right) = \max_{(a_t)} \min_{(\lambda_t)} L\left((\lambda_t), (a_t)\right).$$

- Existence of a saddle point with $(\lambda_t)$ in a nice space is a bit of a complicated problem, but let's assume that there is one

# Recursive formulation

We can construct *co-state* variables from the Lagrangian multipliers

$$\gamma' = \gamma + \lambda.$$

The co-state variable summarizes the impact of the past incentive and the initial participation constraints.

$$F(\gamma, x, s) = \sup_{a \in A} \inf_{\lambda \geq 0}$$
$$\left[ r(x, a, s) + \gamma g(x, a, s) + \lambda (g(x, a, s) - \underline{G}) + \beta E_s F(\gamma + \lambda, x', s') \right]$$
subject to
$$x' = \zeta(x, a, s)$$
$$p(x, a, s) \geq 0.$$

Key is that this problem does not have the future in its constraints and hence is recursive

# The role of strict concavity

Suppose we want to solve the following problem

$$V = \max_{(a_1,a_2)\in[0,1]^2} -a_1 - a_2 \text{ subject to}$$
$$a_1 + a_2 \geq 1$$

The value of the corresponding Lagrangian can be obtained recursively (or by backward induction). We can write

$$\begin{aligned} V_2(\lambda) &= \max_{a_2\in[0,1]} (-a_2 + \lambda a_2) \\ V_1 &= \max_{a_1\in[0,1]} \min_{\lambda\geq 0} (-a_1 + \lambda(a_1 - 1) + V_2(\lambda)) \end{aligned}$$

and get the correct value of the saddle point

$$V_1 = \max_{(a_1,a_2)\in[0,1]^2} \min_{\lambda\geq 0} (-a_1 - a_2) + \lambda(a_1 + a_2 - 1).$$

# The role of strict concavity (2)

However, one cannot recover the correct policies from this if one takes the Lagrange multiplier as a state variable: The correspondence solving the problem in the second period is given by

$$a_2(\lambda) = \arg \max_{a_2 \in [0,1]} (-a_2 + \lambda a_2) = \begin{cases} 0 & \lambda < 1 \\ [0,1] & \lambda = 1 \\ 1 & \lambda > 1. \end{cases}$$

So clearly, one element of the recursive problem's argmax is $a_1 = 1/2$, $\lambda = 1$ and $a_2 = 0$. But this is not a feasible point. Another element of the recursive problem's solution is $a_1 = 1/2, \lambda = 1$ and $a_2 = 1$. This is a feasible point, but obviously suboptimal.

# Example

- We assume that the action $a$ is bounded between $\varepsilon$ and $1-\varepsilon$, where $\varepsilon$ is a small number that serves to bound the payoffs of the agent and the principal.
- The agent each period draws an outside opportunity.
- Assume that in every period there are just two states of the world $s \in \{l, h\}$ which are i.i.d. and equi-probable.
- The conditional opportunity which goes into the incentive constraint (IC) depends on the shock,

$$\bar{g}_1(l) = \frac{1}{1-\beta}\log(\epsilon)$$

and

$$\bar{g}_1(h) = \frac{1}{1-\beta}\log(2/3).$$

The IC cannot bind in state $l$ since $\varepsilon$ is the lowest possible transfer.

# Solve the Example

Letting $\gamma$ be the state variable we get:

$$\begin{aligned} F(\gamma, s) &= \max_{a \in A} \min_{\lambda \geq 0} \log(a) + \gamma \log(1-a) + \lambda \left[\log(1-a) - \bar{g}_1(s)\right] \\ &\quad + \beta E_s F(\gamma + \lambda, \tilde{s}), \end{aligned}$$

where for notational simplicity we have consider only deterministic action choices.

The first order conditions with respect to $a$ is simply

$$\frac{1}{a} - \gamma \frac{1}{1-a} - \lambda \frac{1}{1-a} = 0,$$

and randomization is degenerate.

Nice feature is that we're working simply with the conditional problem.

# Solve the Example

To verify this works need $\gamma + \lambda$ to yield payoff to agent

$$\log(1-a) + \beta G(a,s) = \bar{g}_1(s)$$

if $\lambda(s) > 0$.
From

$$\begin{aligned} F(\gamma, s) &= \max_{a \in A} \min_{\lambda \geq 0} \log(a) + \gamma \log(1-a) + \lambda \left[\log(1-a) - \bar{g}_1(s)\right] \\ &\quad + \beta E_s F(\gamma + \lambda, \tilde{s}), \end{aligned}$$

so

$$\frac{dF(\gamma, s)}{d\lambda} = \left[\log(1-a) - \bar{g}_1(s)\right] + \frac{dE_s F(\gamma + \lambda, \tilde{s})}{d\lambda}.$$

Note that $a$ uniquely determined by $\gamma + \lambda$ and

$$\text{Envelope condition} \quad : \quad \frac{dF(\gamma, s)}{d\gamma} = \log(1-a) + \beta \frac{dE_s F(\gamma + \lambda, \tilde{s})}{d\gamma} = G$$

$$\text{Hence } G(a,s) = \frac{dE_s F(\gamma + \lambda, \tilde{s})}{d\lambda}.$$

# Reinforcement learning

- Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.
- Model free dynamic programming
- Interesting to economists?

# Reinforcement learning and dynamic programming

- For interesting problems both approaches must use compact representations relying on function approximation.
- This challenge was already recognized while the first DP techniques were being developed. However, it has only been in recent years ? and largely in correlation with the advance of RL ? that approximation-based methods have grown in diversity, maturity, and efficiency, enabling RL and DP to scale up to realistic problems.
- Deep RL...
- Today: TD-learning

# FDP: Value function iteration

- State space is $X = \{1, \ldots, S\}$, action sets are $A(x)$ finite sets, $c(x, a)$ finite for all $x, a \in A(x)$. Transition probabilities are $\pi(x'|x, a)$. $\beta < 1$
- For any $v \in \mathbb{R}^S$ define $Tv \in \mathbb{R}^S$ by

$$Tv(x) = \min_{a \in A(x)} c(x, a) + \beta \sum_{x'} \pi(x'|x, a) v(x')$$

- Given and $v_0 \in \mathbb{R}^S$ value function iteration computes

$$v^* = \lim_{n \to \infty} T^n v_0$$

# Q-Value

- Same setup, define policy $\alpha \in \mathbb{R}^S$, $\alpha(x) \in A(x)$. Define

$$v_\alpha(x) = E_x^\alpha \sum_{t=0}^{\infty} \beta^t c(x_t, \alpha(x_t))$$

Define

$$Q_\alpha(x, a) = c(x, a) + \beta \sum_{x'} \pi(x'|x) v_\alpha(x')$$

- Bellman equation

$$Q(x, a) = c(x, a) + \beta \sum_{x'} \pi(x'|x) \min_{a' \in A(x')} Q(x', a')$$

- So what?

# Q-Learning

- Suppose we want to find optimal Q-value function but do not know the state transition or the cost function
- See costs and new state when we do an action
- Initialize $Q^0(.,.) = 0$. Take a sequence of $(\alpha_t)$ with $\alpha_t \to 0$ as $t \to \infty$ as the learning rates.
- Define

$$Q^{t+1}(x_t, a_t) = (1-\alpha_t)Q^t(x_t, a_t) + \alpha_t \left( c(x_t, a_t) + \beta \max_{a \in A(x_{t+1})} Q^t(x_{t+1}, a) \right)$$

- How is $a_t$ chosen?

# The role of exploration vs exploitation

- Often use $\epsilon$-greedy exploration...
- Take sequence $(\epsilon_t)$ $\epsilon_t \to 0$ as $t \to \infty$ and choose $a_t$ randomly over all $a \in A(x_t)$ with probability $\epsilon_t$ and choose $a_t \in \arg\max_{a \in A(x_t)} Q^t(x_t, a)$ otherwise.
- If in the process every state-action pair is visited inifnitely often as $t \to \infty$ then $Q^t \to Q^*$, a solution of the Bellman equation.

# Economic application

- For estimation: Some structural parameters of $c(.)$ must be found along the way
- For learning: Agent does not know the transition probabilities for shocks. Learns and learns to optimize at the same time...
- Check out Marlon's Python code