EGMn

The Sequential Endogenous Grid Method

Alan E. Lujan Solis

September 29, 2023

Motivation

. . .

- Structural Economics for modeling **decision-making under uncertainty**
  - household: consumption, savings, labor, portfolio, retirement
  - firms: production, investment, hiring, entry/exit
  - governments: fiscal and monetary policy, taxation, redistribution
  - interdisciplinary: climate change, public health, education, etc.

. . .

- Structural modeling is **hard**
  - modern economics requires solving complex problems
  - with many state variables, many decisions, and non-convexities
  - computationally challenging and time-consuming

Outline

- Dynamic Programming
  - The Endogenous Grid Method
  - The **Sequential** Endogenous Grid Method
- Functional Approximation
  - Interpolation on different spaces/dimensions
  - Conventional techniques are **insufficient** for complex problems
- Machine Learning in Economics

  - Neural Nets as **function approximators**
  - The Deep Learning Revolution
  - **Gaussian Process** Regression

- Conclusion
  - Computational Economics solving increasingly complex problems
  - **Econ-ARK** provides **open source** tools for researchers

Dynamic Programming

A simple consumption-savings problem

. . .

Agent maximizes present discounted value (PDV) of lifetime utility

$$\max_{c_t} \sum_{t=0}^{\infty} \beta^t \mathrm{u}(c_t) \tag{1}$$

. . .

Recursive Bellman equation

$$\begin{aligned} v_t(m_t) &= \max_{c_t} \mathrm{u}(c_t) + \beta \mathbb{E}_t \left[v_{t+1}(m_{t+1}) \right] \\ \text{s.t.} \quad & 0 < c_t \leq m_t \\ a_t &= m_t - c_t \\ m_{t+1} &= \mathsf{R} a_t + \theta_{t+1} \end{aligned} \tag{2}$$

A simple consumption-savings problem

Recursive Bellman equation

$$\begin{aligned} v_t(m_t) &= \max_{c_t} \mathrm{u}(c_t) + \beta \mathbb{E}_t \left[v_{t+1}(m_{t+1}) \right] \\ \text{s.t.} \quad & 0 < c_t \leq m_t \\ a_t &= m_t - c_t \\ m_{t+1} &= \mathsf{R} a_t + \theta_{t+1} \end{aligned} \tag{3}$$

How do we solve this problem?

- **Value Function Iteration** (VFI)
  - Discretize state space (interpolation)
  - Grid search optimization (brute force, iterative)

The Endogenous Grid Method by Carroll (2006)

. . .

$$v_t(m_t) = \max_{c_t} \mathrm{u}(c_t) + \beta \mathbb{E}_t \left[v_{t+1}(\mathsf{R}(m_t - c_t) + \theta_{t+1}) \right] \tag{4}$$

. . .

$$u'(c_t) - \beta \mathsf{R} \mathbb{E}_t \left[v'_{t+1}(\mathsf{R}(m_t - c_t) + \theta_{t+1}) \right] = 0 \tag{5}$$

. . .

$$c_t = \mathrm{u}'^{-1} \left(\beta \mathsf{R} \mathbb{E}_t \left[v'_{t+1}(\mathsf{R} a_t + \theta_{t+1}) \right] \right) \tag{6}$$

. . .

Contribution:

- Simple
  - **Inverted Euler** equation
- Fast
  - No **root-finding** or **grid search** optimization required
- Efficient
  - Finds **exact solution** at each gridpoint

Limitations of EGM

. . .

- **One-dimensional** problems/sub-problems (nested)
  - (GEGM) Barillas and Fernández-Villaverde (2007)
  - (NEGM) Druedahl (2021)
- Can result in **non-rectangular grids**
  - (Curvilinear) White (2015)
  - (Triangular) Ludwig and Schön (2018)
- **Non-convexities** (discrete choices) can be problematic
  - (DCEGM) Iskhakov et al. (2017)
  - (G2EGM) Druedahl and Jørgensen (2017)

EGMn: The Sequential Endogenous Grid Method

. . .

- **Insight**: Problems in which agent makes several **simultaneous choices** can be decomposed into **sequence of problems**
- **Problem**: Rectilinear exogenous grid results in **unstructured** endogenous grid
- **Solution**: Using machine learning to **interpolate** on unstructured grids

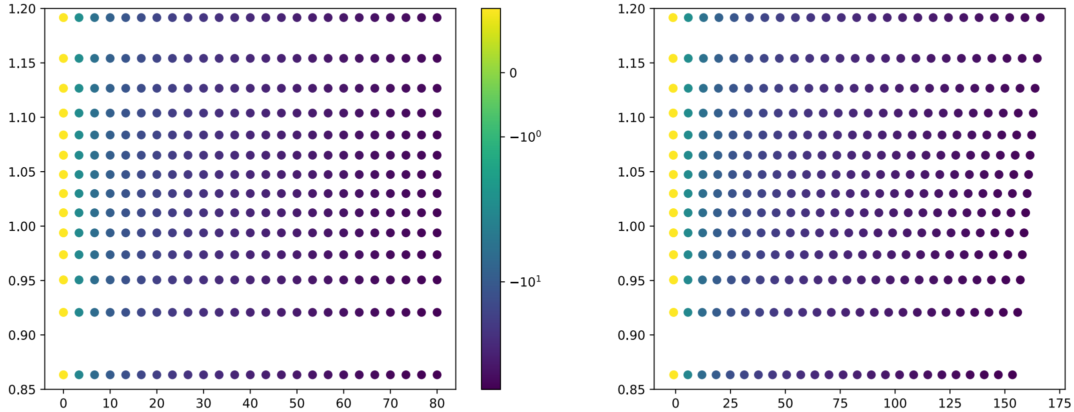
. . .

Contribution:

- **Simple, Fast, Efficient**
  - Inherits properties of EGM
- **Multi-dimensional**
  - Can be used for problems with multiple state variables and decisions
- **Cutting-edge**
  - Functional approximation and uncertainty quantification approach using Gaussian Process Regression

Exogenous Rectangular Grid

Endogenous Curvilinear Grid



A more complex problem

. . .

Consumption - Pension Deposit Problem as in Druedahl and Jørgensen (2017)

$$\begin{aligned} \mathrm{v}_t(m_t, n_t) &= \max_{c_t, d_t} u(c_t) + \beta \mathbb{E}_t \left[\mathrm{v}_{t+1}(m_{t+1}, n_{t+1}) \right] \\ \text{s.t.} \quad c_t &> 0, \quad d_t \geq 0 \\ a_t &= m_t - c_t - d_t \\ b_t &= n_t + d_t + g(d_t) \\ m_{t+1} &= a_t \mathsf{R} + \theta_{t+1} \\ n_{t+1} &= b_t \mathbf{R}_{t+1} \end{aligned} \tag{7}$$

. . .

where

$$\mathrm{u}(c) = \frac{c^{1-\rho}}{1-\rho} \qquad \text{and} \qquad \mathrm{g}(d) = \chi \log(1 + d). \tag{8}$$

is a tax-advantaged premium on pension contributions.

G2EGM from Druedahl and Jørgensen (2017)

. . .

- If we try to use EGM:
  - 2 first order conditions
  - multiple constraints difficult to handle
  - segments: combinations of first order conditions and constraints
  - 2^d segments where d is number of control variables
  - requires local triangulation interpolation

Breaking up the problem makes it easier

. . .

Consider the problem of a consumer who chooses how much to put into a pension account:

$$
\begin{aligned}
\mathrm{v}_t(m_t, n_t) = \max_{d_t} \tilde{\mathfrak{v}}_t(l_t, b_t)& \\
\text{s.t.} \quad 0 \leq d_t \leq m_t& \\
l_t = m_t - d_t& \\
b_t = n_t + d_t + g(d_t)&
\end{aligned}
\tag{9}
$$

. . .

After, the consumer chooses how much to consume out of liquid savings:

$$
\begin{aligned}
\tilde{\mathfrak{v}}_t(l_t, b_t) = \max_{c_t} u(c_t) + \beta \mathrm{w}_t(a_t, b_t)& \\
\text{s.t.} \quad 0 < c_t \leq m_t& \\
a_t = l_t - c_t&
\end{aligned}
\tag{10}
$$

Breaking up the problem makes it easier

Consider the problem of a consumer who chooses how much to put into a pension account:

$$
\begin{aligned}
\mathrm{v}_t(m_t, n_t) &= \max_{d_t} \tilde{\mathfrak{v}}_t(l_t, b_t) \\
&\text{s.t.} \quad 0 \leq d_t \leq m_t \\
l_t &= m_t - d_t \\
b_t &= n_t + d_t + g(d_t)
\end{aligned} \tag{11}
$$

After, the consumer chooses how much to consume out of liquid savings:

$$
\begin{aligned}
\tilde{\mathfrak{v}}_t(l_t, b_t) &= \max_{c_t} u(c_t) + \beta \mathrm{w}_t(a_t, b_t) \\
&\text{s.t.} \quad 0 < c_t \leq m_t \\
a_t &= l_t - c_t
\end{aligned} \tag{12}
$$

And the post-decision value function is defined as:

$$
\begin{aligned}
\mathrm{w}_t(a_t, b_t) &= \mathbb{E}_t \left[\mathrm{v}_{t+1}(m_{t+1}, n_{t+1}) \right] \\
&\text{s.t.} \\
m_{t+1} &= a_t \mathsf{R} + \theta_{t+1} \\
n_{t+1} &= b_t \mathbf{R}_{t+1}
\end{aligned} \tag{13}
$$

Breaking up the problem makes it easier

Consider the problem of a consumer who chooses how much to put into a pension account:

$$
\begin{aligned}
\mathrm{v}_t(m_t, n_t) &= \max_{d_t} \tilde{\mathfrak{v}}_t(l_t, b_t) \\
&\text{s.t.} \quad 0 \leq d_t \leq m_t \\
l_t &= m_t - d_t \\
b_t &= n_t + d_t + g(d_t)
\end{aligned} \tag{14}
$$

After, the consumer chooses how much to consume out of liquid savings:

$$
\begin{aligned}
\tilde{\mathfrak{v}}_t(l_t, b_t) &= \max_{c_t} u(c_t) + \beta \mathrm{w}_t(a_t, b_t) \\
&\text{s.t.} \quad 0 < c_t \leq m_t \\
a_t &= l_t - c_t
\end{aligned} \tag{15}
$$

And the post-decision value function is defined as:

$$
\begin{aligned}
\mathrm{w}_t(a_t, b_t) &= \mathbb{E}_t \left[\mathrm{v}_{t+1}(m_{t+1}, n_{t+1}) \right] \\
&\text{s.t.} \\
m_{t+1} &= a_t \mathsf{R} + \theta_{t+1} \\
n_{t+1} &= b_t \mathbf{R}_{t+1}
\end{aligned} \tag{16}
$$

Steps:

1. Compute $\mathrm{w}_t(a_t, b_t)$
2. Solve consumption problem (EGM)
3. Solve pension problem (EGM, again)
4. Done!

Solving the pension problem

. . .

The pension problem, more compactly

$$
\mathrm{v}_t(m_t, n_t) = \max_{d_t} \tilde{\mathfrak{v}}_t(m_t - d_t, n_t + d_t + \mathrm{g}(d_t)) \tag{17}
$$

. . .

Interior solution must satisfy the first-order condition:

$$
\mathrm{g}'(d_t) = \frac{\tilde{\mathfrak{v}}_t^l(l_t, b_t)}{\tilde{\mathfrak{v}}_t^b(l_t, b_t)} - 1 \tag{18}
$$

. . .

Inverting, we can obtain the optimal choice of d_t:

$$
\mathfrak{d}_t(l_t, b_t) = \mathrm{g}'^{-1} \left(\frac{\tilde{\mathfrak{v}}_t^l(l_t, b_t)}{\tilde{\mathfrak{v}}_t^b(l_t, b_t)} - 1 \right) \tag{19}
$$

Solving the pension problem

Inverting, we can obtain the optimal choice of d_t:

$$\mathfrak{d}_t(l_t, b_t) = \mathrm{g}'^{-1}\left(\frac{\tilde{\mathfrak{v}}_t^l(l_t, b_t)}{\tilde{\mathfrak{v}}_t^b(l_t, b_t)} - 1\right) \tag{20}$$

Using resource constraints we obtain endogenous grids:

$$\mathfrak{n}_t(l_t, b_t) = b_t - \mathfrak{d}_t(l_t, b_t) - \mathrm{g}(\mathfrak{d}_t(l_t, b_t))\mathfrak{m}_t(l_t, b_t) = l_t + \mathfrak{d}_t(l_t, b_t) \tag{21}$$

. . .

Now we have the triple $\{\mathfrak{m}_t, \mathfrak{n}_t, \mathfrak{d}_t\}$ where $\mathfrak{d}_t$ is the unconstrained approx. of optimal deposit for each $(\mathfrak{m}_t, \mathfrak{n}_t)$ corresponding to each (l_t, b_t). Generally, we can construct an interpolator as follows:

$$\hat{d}_t(\mathfrak{m}_t, \mathfrak{n}_t) = \begin{cases} 0 & \text{if } \mathfrak{d}_t < 0 \\ \mathfrak{d}_t & \text{if } 0 \le \mathfrak{d}_t \le \mathfrak{m}_t \\ \mathfrak{m}_t & \text{if } \mathfrak{d}_t > \mathfrak{m}_t \end{cases} \tag{22}$$

Unstructured Grids

. . .

Problem: **Rectilinear** exogenous grid results in **unstructured** endogenous grid

Exogenous Rectangular Grid

Endogenous Unstructured Grid

How do we **interpolate** on this grid?

Functional Approximation

Linear Interpolation on a Uniform Grid

videos/LinearInterpolationUniform.mp4

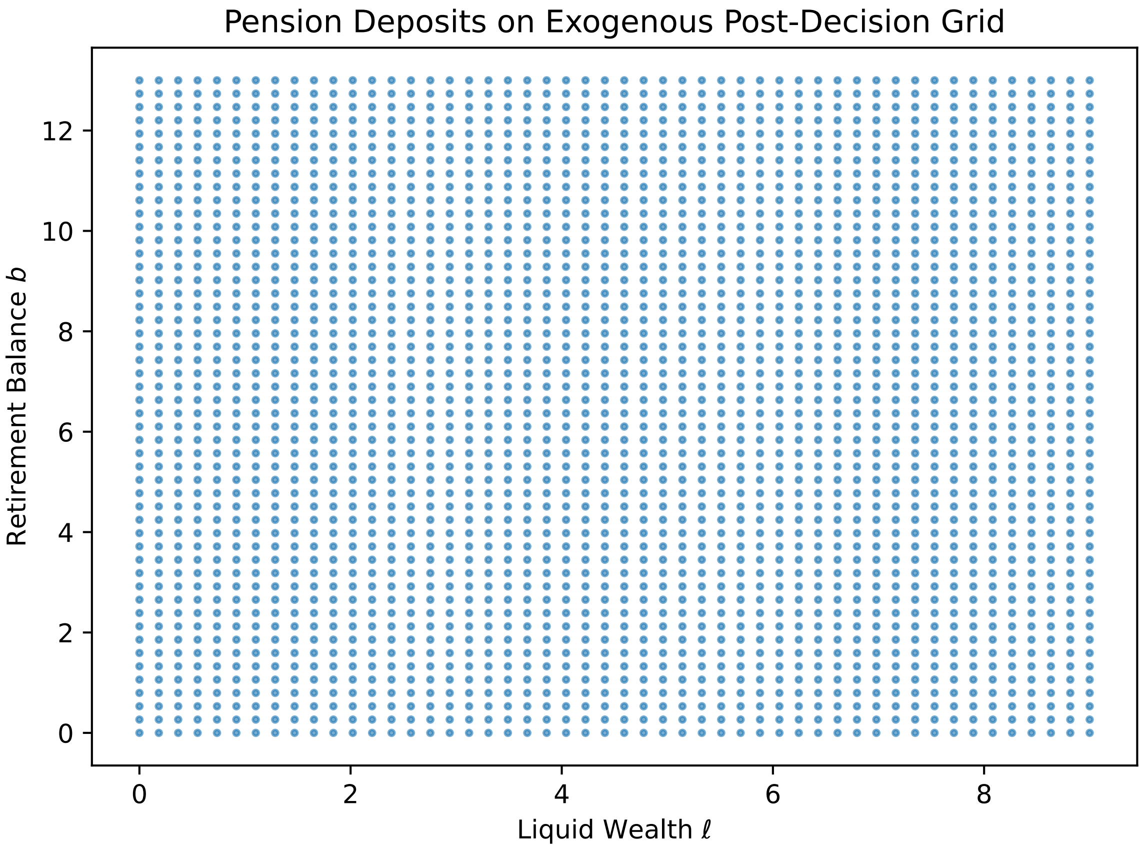


Figure 1: Sparse Pension Exogenous Grid

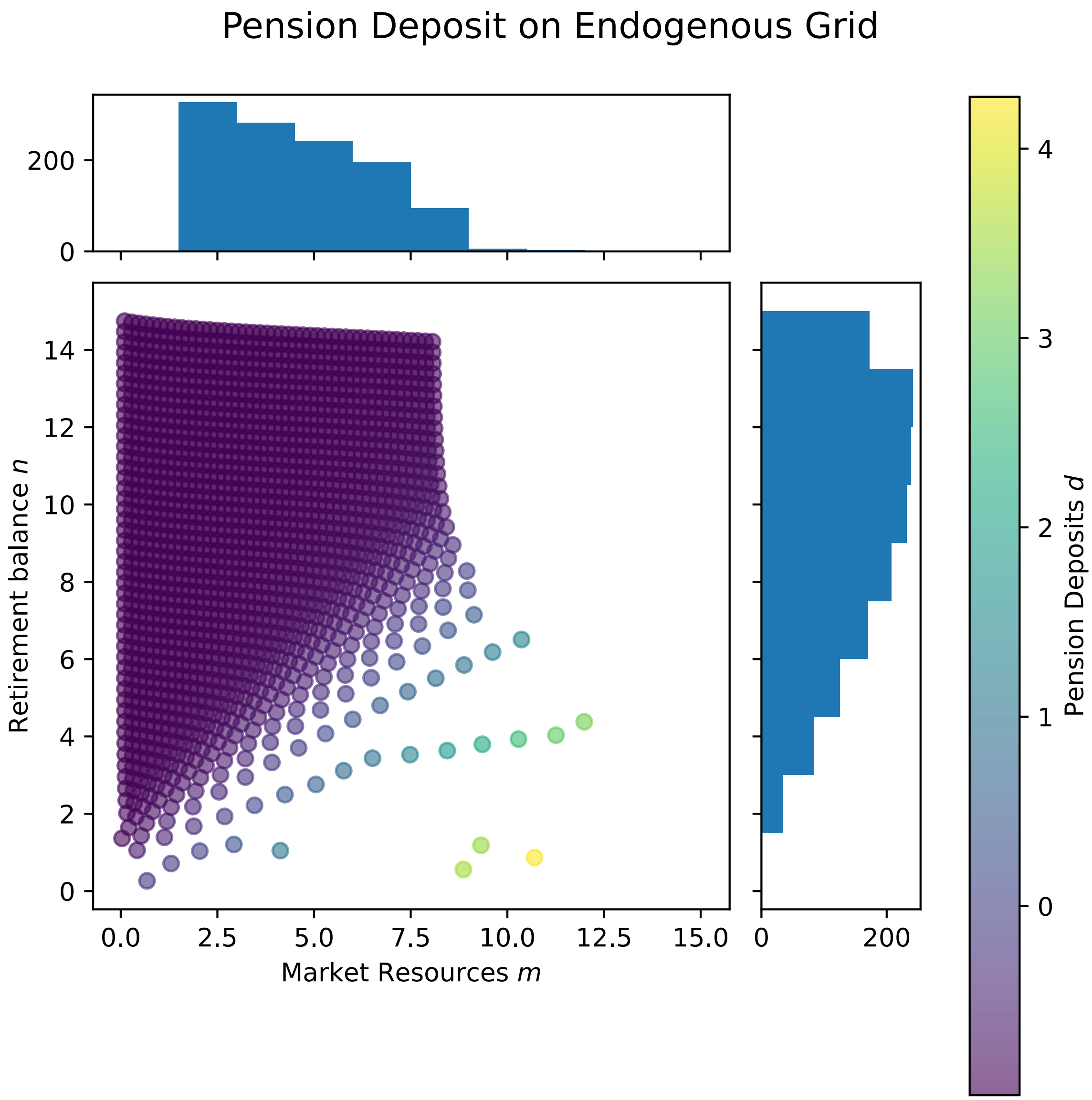


Figure 2: Unstructured Pension Endogenous Grid

Linear Interpolation on a Non-linear Grid

videos/LinearInterpolationGeometric.mp4

Bilinear Interpolation

videos/BilinearInterpolation.mp4

Curvilinear (Warped) Grid Interpolation

videos/CurvilinearInterpolation.mp4

See: White (2015)

What about Unstructured Grids?

videos/UnstructuredGrid.mp4

See: Ludwig and Schön (2018)

Machine Learning in Economics

Artificial Neural Networks

. . .

Artificial Neural Networks

- Based on biological neural pathways (neurons in a brain)
- Learns function $f(X) : R^n \to R^m$
- Consists of
  - input (features) X
  - hidden layers $g(\cdots)$
  - output (target) $y = f(X)$
- Hidden layers can have many nodes
- Neural nets can have many hidden layers (deep learning)

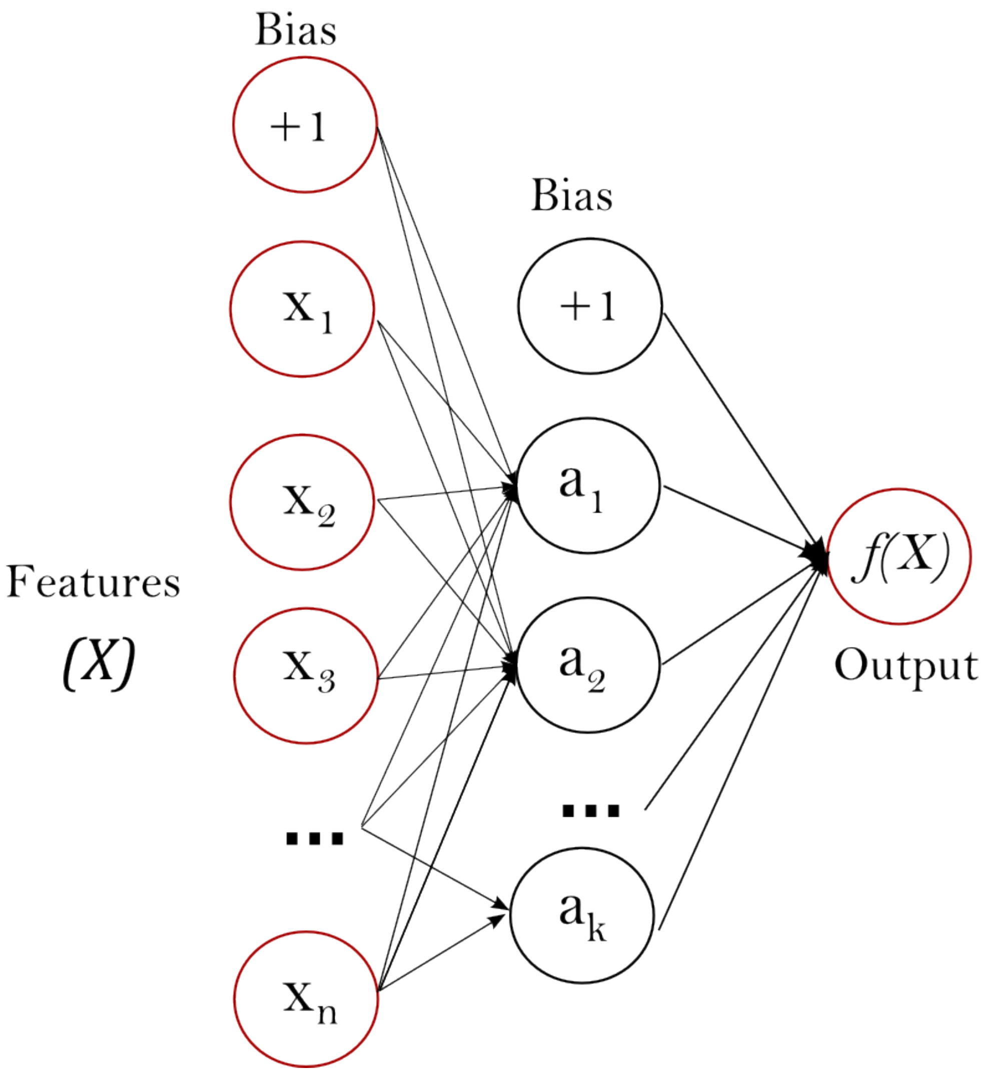


Figure 3: Figure 1: ANN (Source: scikit-learn.org)

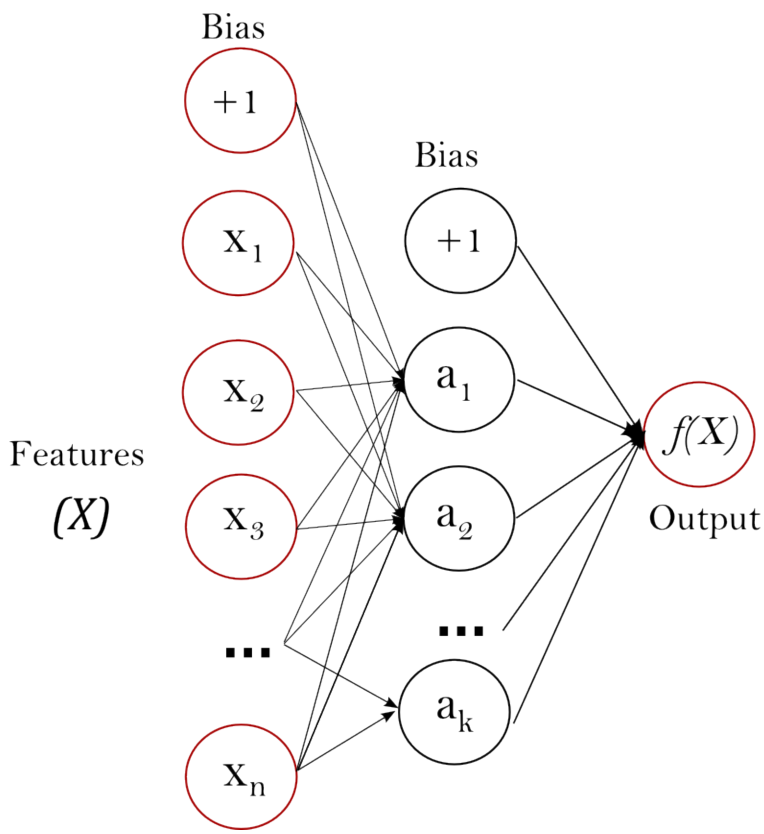


Figure 4: Figure 1: ANN (Source: scikit-learn.org)

A single neuron, and a bit of math

. . .

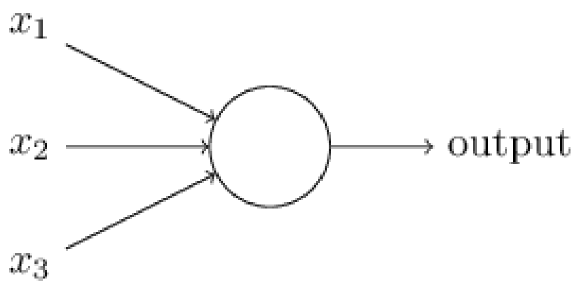


Figure 5: Figure 2: Perceptron

$$y = g(w_0 + \sum_{i=1}^{n} w_i x_i) = g(w_0 + \mathbf{x}'\mathbf{w}) \tag{23}$$

A single neuron, and a bit of math

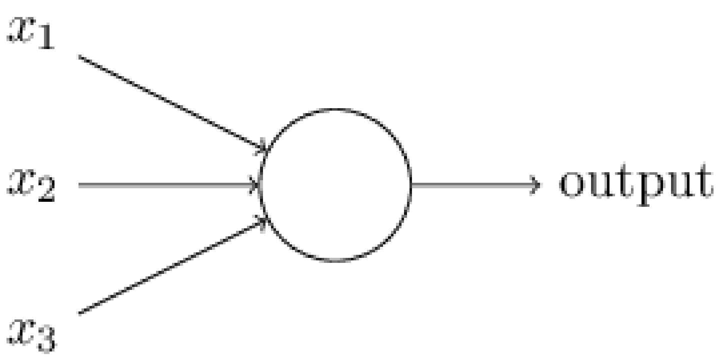


Figure 6: Figure 2: Perceptron

$$y = g(w_0 + \sum_{i=1}^{n} w_i x_i) = g(w_0 + \mathbf{x}'\mathbf{w}) \tag{24}$$

. . .

- y is the output or target
- x_i are the inputs or features
- w_0 is the bias

- w_i are the weights
- $g(\cdot)$ is the activation function (non-linear)

$$g(z) = \frac{1}{1 + e^{-z}} \tag{25}$$

- usually a sigmoid, but there are many others

The Deep Learning Revolution

. . .

- Most of these ideas are not new
  - Perceptron (1957)
  - Deep Learning (1965)
  - Stochastic Gradient Descent (1967)

. . .

- What changed?
  - **Big data** (more data)
  - More computing power (**GPUs**, TPUs, etc.)
  - **Algorithmic** innovations (ReLU, Adam, regularization, etc.)
  - Better and **open source** software (scikit-learn, TensorFlow, PyTorch, etc.)

Gaussian Process Regression

. . .

A Gaussian Process is a **probability distribution over functions**

$$\begin{aligned} f(x) &\sim \mathcal{GP}(m(x), k(x, x')) \\ \text{where} \quad m(x) &= \mathbb{E}[f(x)] \\ \text{and} \quad k(x, x') &= \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \end{aligned} \tag{26}$$

. . .

A Gaussian Process **Regression** is used to find the function that **best fits** a set of data points

$$\mathbb{P}(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{K}) \tag{27}$$

. . .

I use standard covariance function, exploring alternatives is an active area of research

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)'(\mathbf{x}_i - \mathbf{x}_j)\right). \tag{28}$$

. . .

Universal Approximation Theorem: A single hidden-layer ANN can **approximate** any continuous function **arbitrarily closely** as the number of neurons in the hidden layer **increases**. Notably, a Gaussian Process (GP) can be viewed as the **limit** of a single hidden-layer ANN with an **infinite** number of neurons (**infinite width**).

An example

. . .

Consider the true function $f(x) = x\cos(1.5x)$ sampled at random points

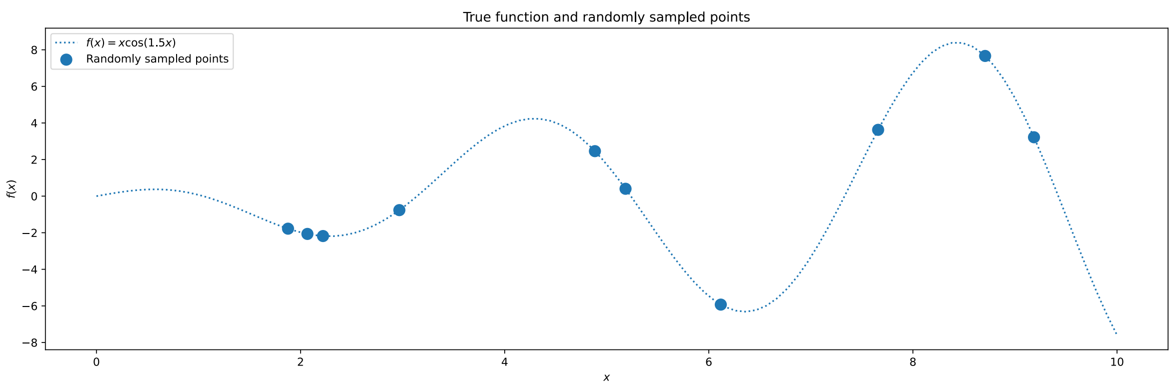


Figure 7: True Function

An example

. . .

A random sample of the GP posterior distribution of functions

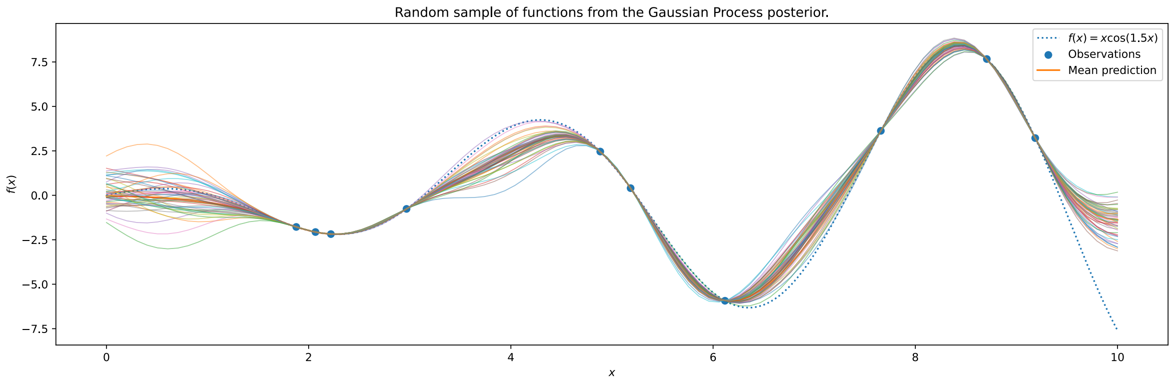
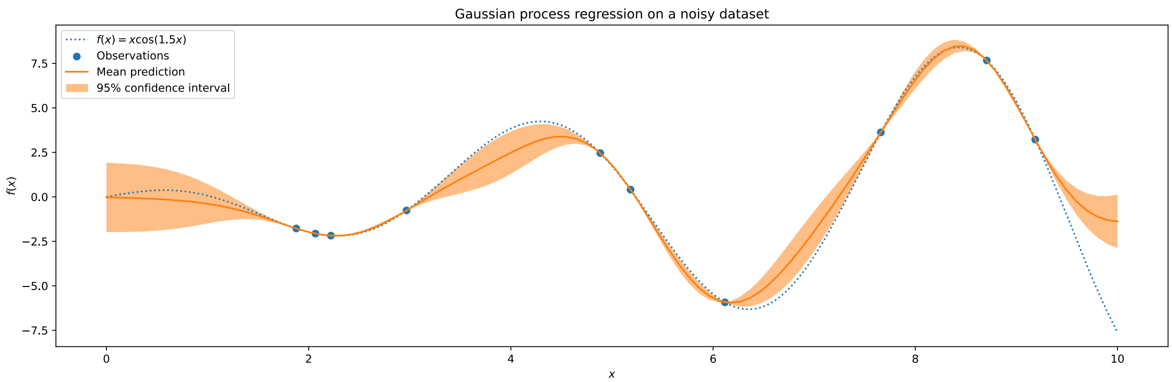


Figure 8: Posterior Sample

An example

. . .

Gaussian Process Regression finds the function that best fits the data



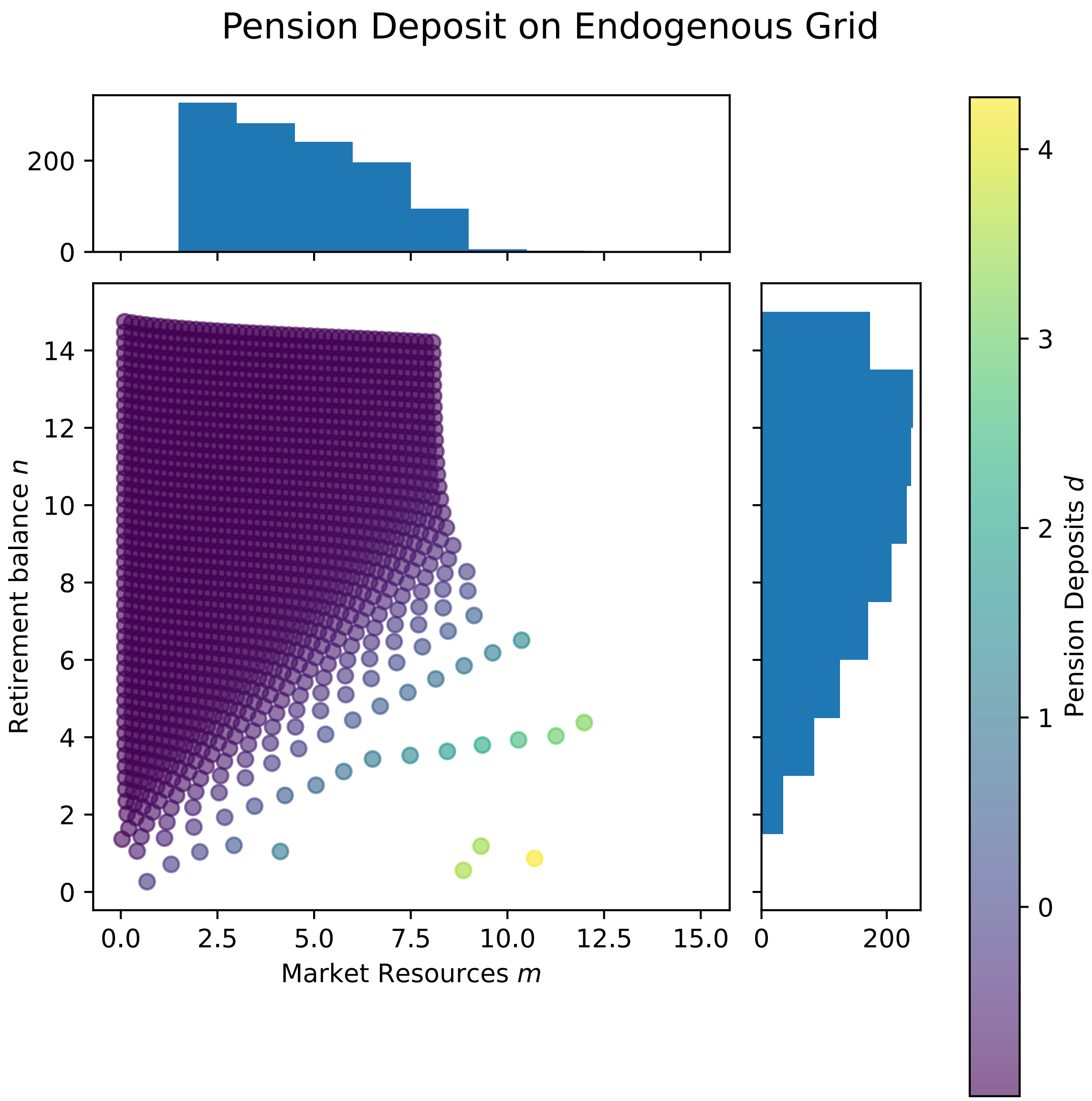
. . .

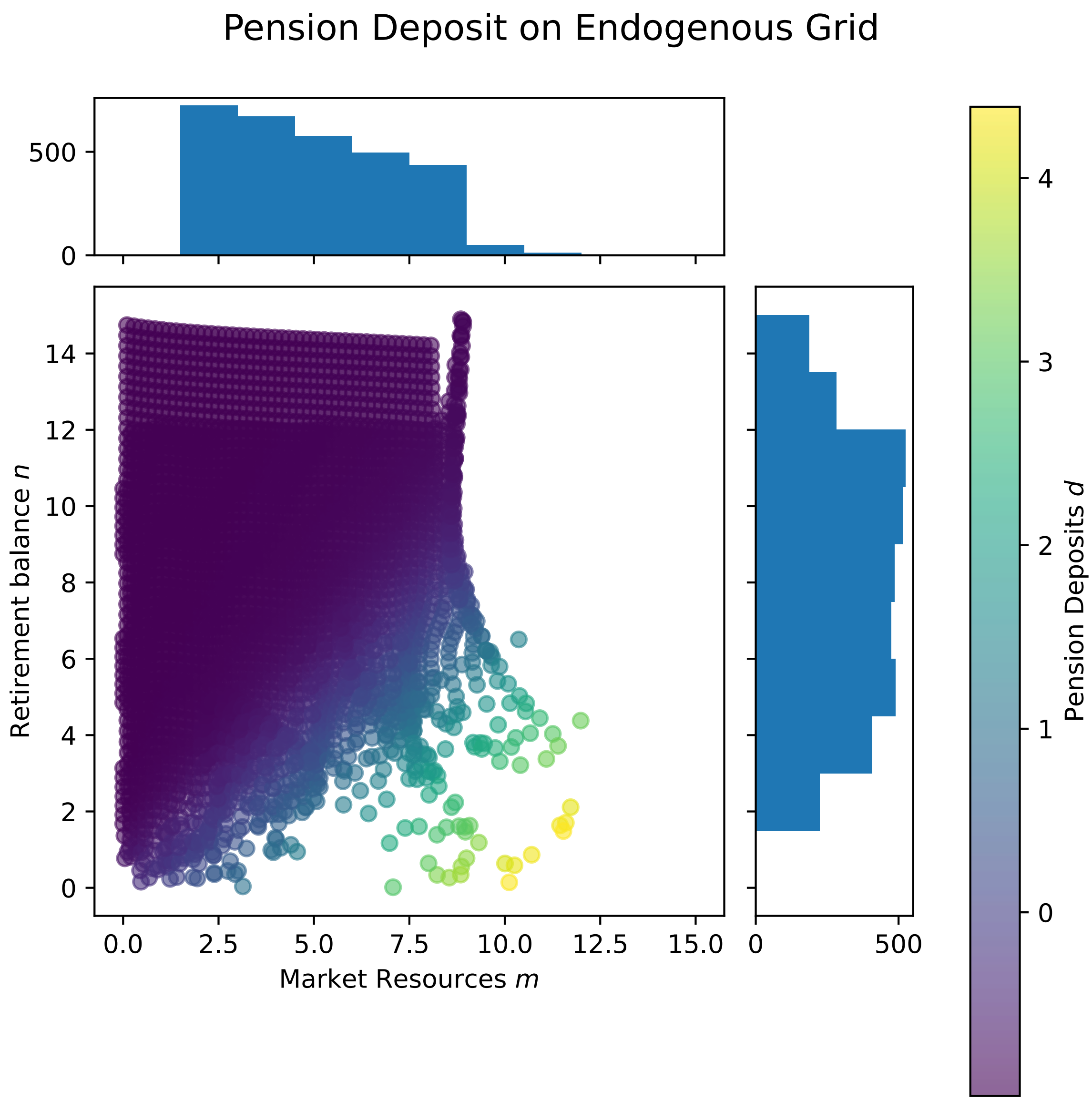
- **Gaussian Process Regression** gives us
  - **Mean** function of the posterior distribution
  - **Uncertainty quantification** of the mean function
  - Can be useful to predict ex-post where we might need **more points**

Back to the model

. . .

Second Stage Pension Endogenous Grid





Some Results

. . .

Consumption Function

Deposit Function

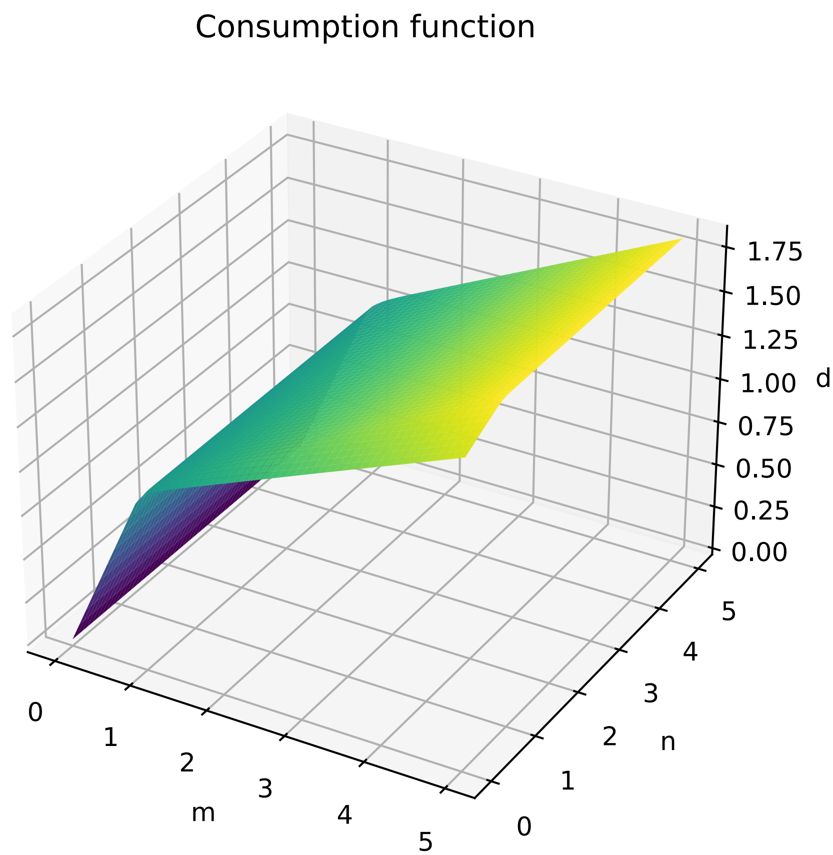


Figure 9: Pension Consumption Function

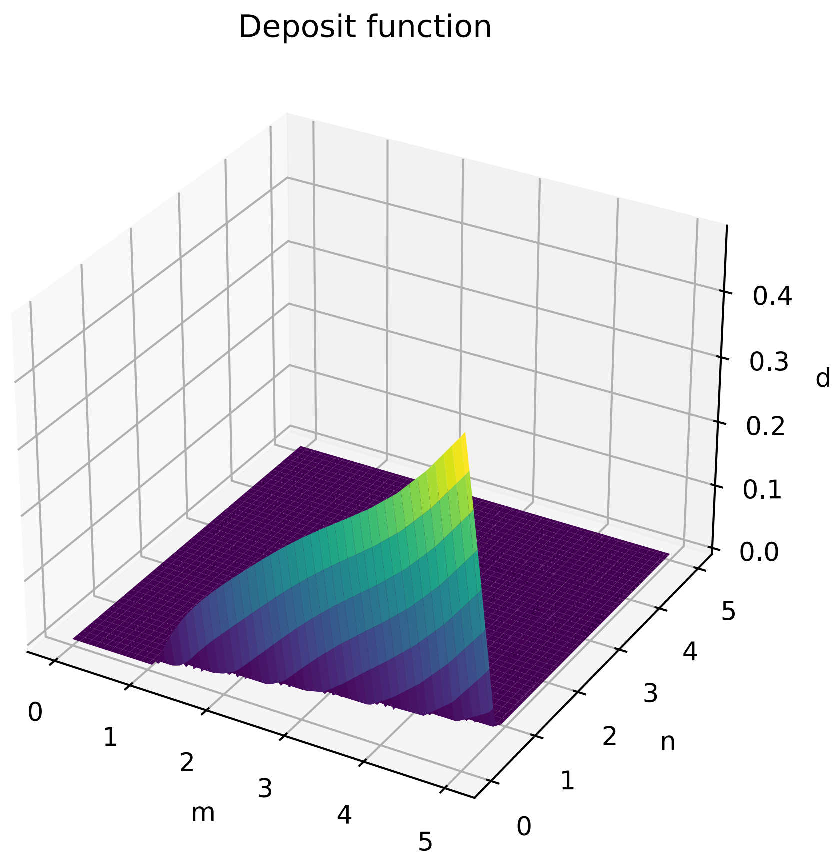


Figure 10: Penstion Deposit Function

Conclusion

Conditions for using Sequential EGM

. . .

- Model must be
  - concave
  - differentiable
  - continuous
  - separable

. . .

Need an **additional** function to exploit **invertibility**

. . .

Examples in this paper:

- Separable utility function
  - $\mathrm{u}(c, z) = \mathrm{u}(c) + h(z)$
- Continuous and differentiable transition
  - $b_t = n_t + d_t + g(d_t)$

Resources

. . .

- An Introduction to Statistical Learning `statlearning.com`
- Neural Networks and Deep Learning `neuralnetworksanddeeplearning.com`
- Deep Learning `deeplearningbook.org`
- Probabilistic machine learning `probml.github.io/pml-book`
- A Neural Network Playground `playground.tensorflow.org`
- The Gaussian Process Web Site `gaussianprocess.org`
- A Visual Exploration of Gaussian Processes `distill.pub/2019`
- Interactive Gaussian Process Visualization `http://www.infinitecuriosity.org/vizgp`

Thank you!

. . .

Powered by Econ-ARK

engine: github.com/econ-ark/HARK

code: github.com/alanlujan91/SequentialEGM

website: alanlujan91.github.io/SequentialEGM/egmn

References

Barillas, Francisco, and Jesús Fernández-Villaverde. 2007. "A Generalization of the Endogenous Grid Method." *Journal of Economic Dynamics & Control* 31 (8): 2698–2712. https://doi.org/10.1016/j.jedc.2006.08.005.

Carroll, Christopher D. 2006. "The Method of Endogenous Gridpoints for Solving Dynamic Stochastic Optimization Problems." *Economics Letters* 91 (3): 312–20. https://doi.org/10.1016/j.econlet.2005.09.013.

Druedahl, Jeppe. 2021. "A Guide on Solving Non-Convex consumption-saving Models." *Computational Economics* 58 (3): 747–75. https://doi.org/10.1007/s10614-020-10045-x.

Druedahl, Jeppe, and Thomas Høgholm Jørgensen. 2017. "A General Endogenous Grid Method for Multi-Dimensional Models with Non-Convexities and Constraints." *Journal of*

Economic Dynamics & Control 74 (January): 87–107. https://doi.org/10.1016/j.jedc.2016.11.005.

Iskhakov, Fedor, Thomas H Jørgensen, John Rust, and Bertel Schjerning. 2017. "The Endogenous Grid Method for Discrete-Continuous Dynamic Choice Models with (or Without) Taste Shocks: DC-EGM Method for Dynamic Choice Models." *Quantitative Economics* 8 (2): 317–65. https://doi.org/10.3982/qe643.

Ludwig, Alexander, and Matthias Schön. 2018. "Endogenous Grids in Higher Dimensions: Delaunay Interpolation and Hybrid Methods." *Computational Economics* 51 (3): 463–92. https://doi.org/10.1007/s10614-016-9611-2.

White, Matthew N. 2015. "The Method of Endogenous Gridpoints in Theory and Practice." *Journal of Economic Dynamics & Control* 60 (November): 26–41. https://doi.org/10.1016/j.jedc.2015.08.001.