Part V

Classic Linear Models

CHAPTER

TWENTYSIX

VON NEUMANN GROWTH MODEL (AND A GENERALIZATION)

Contents

- *Von Neumann Growth Model (and a Generalization)*
  - *Notation*
  - *Model Ingredients and Assumptions*
  - *Dynamic Interpretation*
  - *Duality*
  - *Interpretation as Two-player Zero-sum Game*

This lecture uses the class `Neumann` to calculate key objects of a linear growth model of John von Neumann [vN37] that was generalized by Kemeny, Morgenstern and Thompson [KMT56].

Objects of interest are the maximal expansion rate (α), the interest factor (β), the optimal intensities (x), and prices (p).

In addition to watching how the towering mind of John von Neumann formulated an equilibrium model of price and quantity vectors in balanced growth, this lecture shows how fruitfully to employ the following important tools:

- a zero-sum two-player game
- linear programming
- the Perron-Frobenius theorem

We'll begin with some imports:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import solve
from scipy.optimize import fsolve, linprog
from textwrap import dedent
%matplotlib inline

np.set_printoptions(precision=2)
```

The code below provides the `Neumann` class

```
class Neumann(object):

    """
    This class describes the Generalized von Neumann growth model as it was
    discussed in Kemeny et al. (1956, ECTA) and Gale (1960, Chapter 9.5):
```

(continues on next page)

(continued from previous page)

```
    Let:
    n ... number of goods
    m ... number of activities
    A ... input matrix is m-by-n
        a_{i,j} - amount of good j consumed by activity i
    B ... output matrix is m-by-n
        b_{i,j} - amount of good j produced by activity i

    x ... intensity vector (m-vector) with non-negative entries
        x'B - the vector of goods produced
        x'A - the vector of goods consumed
    p ... price vector (n-vector) with non-negative entries
        Bp - the revenue vector for every activity
        Ap - the cost of each activity

    Both A and B have non-negative entries. Moreover, we assume that
    (1) Assumption I (every good which is consumed is also produced):
        for all j, b_{.,j} > 0, i.e. at least one entry is strictly positive
    (2) Assumption II (no free lunch):
        for all i, a_{i,.} > 0, i.e. at least one entry is strictly positive

    Parameters
    ----------
    A : array_like or scalar(float)
        Part of the state transition equation.  It should be `n x n`
    B : array_like or scalar(float)
        Part of the state transition equation.  It should be `n x k`
    """

    def __init__(self, A, B):

        self.A, self.B = list(map(self.convert, (A, B)))
        self.m, self.n = self.A.shape

        # Check if (A, B) satisfy the basic assumptions
        assert self.A.shape == self.B.shape, 'The input and output matrices \
              must have the same dimensions!'
        assert (self.A >= 0).all() and (self.B >= 0).all(), 'The input and \
              output matrices must have only non-negative entries!'

        # (1) Check whether Assumption I is satisfied:
        if (np.sum(B, 0) <= 0).any():
            self.AI = False
        else:
            self.AI = True

        # (2) Check whether Assumption II is satisfied:
        if (np.sum(A, 1) <= 0).any():
            self.AII = False
        else:
            self.AII = True

    def __repr__(self):
        return self.__str__()

    def __str__(self):
```

(continues on next page)

(continued from previous page)

```
        me = """
        Generalized von Neumann expanding model:
          - number of goods           : {n}
          - number of activities      : {m}

        Assumptions:
          - AI:  every column of B has a positive entry    : {AI}
          - AII: every row of A has a positive entry       : {AII}

        """
        # Irreducible                                       : {irr}
        return dedent(me.format(n=self.n, m=self.m,
                                AI=self.AI, AII=self.AII))

    def convert(self, x):
        """
        Convert array_like objects (lists of lists, floats, etc.) into
        well-formed 2D NumPy arrays
        """
        return np.atleast_2d(np.asarray(x))


    def bounds(self):
        """
        Calculate the trivial upper and lower bounds for alpha (expansion rate)
        and beta (interest factor). See the proof of Theorem 9.8 in Gale (1960)
        """

        n, m = self.n, self.m
        A, B = self.A, self.B

        f = lambda α: ((B - α * A) @ np.ones((n, 1))).max()
        g = lambda β: (np.ones((1, m)) @ (B - β * A)).min()

        UB = fsolve(f, 1).item()  # Upper bound for α, β
        LB = fsolve(g, 2).item()  # Lower bound for α, β

        return LB, UB


    def zerosum(self, y, dual=False):
        """
        Given gamma, calculate the value and optimal strategies of a
        two-player zero-sum game given by the matrix

                M(gamma) = B - gamma * A

        Row player maximizing, column player minimizing

        Zero-sum game as an LP (primal --> α)

            max (0', 1) @ (x', v)
            subject to
            [-M', ones(n, 1)] @ (x', v)' <= 0
            (x', v) @ (ones(m, 1), 0) = 1
            (x', v) >= (0', -inf)
```

(continues on next page)

(continued from previous page)

```
        Zero-sum game as an LP (dual --> beta)

            min (0', 1) @ (p', u)
            subject to
            [M, -ones(m, 1)] @ (p', u)' <= 0
            (p', u) @ (ones(n, 1), 0) = 1
            (p', u) >= (0', -inf)

        Outputs:
        --------
        value: scalar
            value of the zero-sum game

        strategy: vector
            if dual = False, it is the intensity vector,
            if dual = True, it is the price vector
        """

        A, B, n, m = self.A, self.B, self.n, self.m
        M = B - y * A

        if dual == False:
            # Solve the primal LP (for details see the description)
            # (1) Define the problem for v as a maximization (linprog minimizes)
            c = np.hstack([np.zeros(m), -1])

            # (2) Add constraints :
            # ... non-negativity constraints
            bounds = tuple(m * [(0, None)] + [(None, None)])
            # ... inequality constraints
            A_iq = np.hstack([-M.T, np.ones((n, 1))])
            b_iq = np.zeros((n, 1))
            # ... normalization
            A_eq = np.hstack([np.ones(m), 0]).reshape(1, m + 1)
            b_eq = 1

            res = linprog(c, A_ub=A_iq, b_ub=b_iq, A_eq=A_eq, b_eq=b_eq,
                          bounds=bounds)

        else:
            # Solve the dual LP (for details see the description)
            # (1) Define the problem for v as a maximization (linprog minimizes)
            c = np.hstack([np.zeros(n), 1])

            # (2) Add constraints :
            # ... non-negativity constraints
            bounds = tuple(n * [(0, None)] + [(None, None)])
            # ... inequality constraints
            A_iq = np.hstack([M, -np.ones((m, 1))])
            b_iq = np.zeros((m, 1))
            # ... normalization
            A_eq = np.hstack([np.ones(n), 0]).reshape(1, n + 1)
            b_eq = 1

            res = linprog(c, A_ub=A_iq, b_ub=b_iq, A_eq=A_eq, b_eq=b_eq,
                          bounds=bounds)
```

(continues on next page)

(continued from previous page)

```
        if res.status != 0:
            print(res.message)

        # Pull out the required quantities
        value = res.x[-1]
        strategy = res.x[:-1]

        return value, strategy


    def expansion(self, tol=1e-8, maxit=1000):
        """
        The algorithm used here is described in Hamburger-Thompson-Weil
        (1967, ECTA). It is based on a simple bisection argument and utilizes
        the idea that for a given γ (= α or β), the matrix "M = B - γ * A"
        defines a two-player zero-sum game, where the optimal strategies are
        the (normalized) intensity and price vector.

        Outputs:
        --------
        alpha: scalar
            optimal expansion rate
        """

        LB, UB = self.bounds()

        for iter in range(maxit):

            γ = (LB + UB) / 2
            ZS = self.zerosum(γ=γ)
            V = ZS[0]      # value of the game with γ

            if V >= 0:
                LB = γ
            else:
                UB = γ

            if abs(UB - LB) < tol:
                γ = (UB + LB) / 2
                x = self.zerosum(γ=γ)[1]
                p = self.zerosum(γ=γ, dual=True)[1]
                break

        return γ, x, p

    def interest(self, tol=1e-8, maxit=1000):
        """
        The algorithm used here is described in Hamburger-Thompson-Weil
        (1967, ECTA). It is based on a simple bisection argument and utilizes
        the idea that for a given gamma (= alpha or beta),
        the matrix "M = B - γ * A" defines a two-player zero-sum game,
        where the optimal strategies are the (normalized) intensity and price
        vector

        Outputs:
        --------
```

(continues on next page)

(continued from previous page)

```
        beta: scalar
            optimal interest rate
        """

        LB, UB = self.bounds()

        for iter in range(maxit):
            y = (LB + UB) / 2
            ZS = self.zerosum(y=y, dual=True)
            V = ZS[0]

            if V > 0:
                LB = y
            else:
                UB = y

            if abs(UB - LB) < tol:
                y = (UB + LB) / 2
                p = self.zerosum(y=y, dual=True)[1]
                x = self.zerosum(y=y)[1]
                break

        return y, x, p
```

26.1 Notation

We use the following notation.

0 denotes a vector of zeros.

We call an n-vector positive and write $x \gg \mathbf{0}$ if $x_i > 0$ for all $i = 1, 2, \dots, n$.

We call a vector non-negative and write $x \geq \mathbf{0}$ if $x_i \geq 0$ for all $i = 1, 2, \dots, n$.

We call a vector semi-positive and written $x > \mathbf{0}$ if $x \geq \mathbf{0}$ and $x \neq \mathbf{0}$.

For two conformable vectors x and y, $x \gg y$, $x \geq y$ and $x > y$ mean $x - y \gg \mathbf{0}$, $x - y \geq \mathbf{0}$, and $x - y > \mathbf{0}$, respectively.

We let all vectors in this lecture be column vectors; x^T denotes the transpose of x (i.e., a row vector).

Let ι_n denote a column vector composed of n ones, i.e. $\iota_n = (1, 1, \dots, 1)^T$.

Let e^i denote a vector (of arbitrary size) containing zeros except for the i th position where it is one.

We denote matrices by capital letters. For an arbitrary matrix A, $a_{i,j}$ represents the entry in its i th row and j th column. $a_{\cdot j}$ and $a_{i \cdot}$ denote the j th column and i th row of A, respectively.

26.2 Model Ingredients and Assumptions

A pair (A, B) of $m \times n$ non-negative matrices defines an economy.

- m is the number of *activities* (or sectors)
- n is the number of *goods* (produced and/or consumed).
- A is called the *input matrix*; $a_{i,j}$ denotes the amount of good j consumed by activity i
- B is called the *output matrix*; $b_{i,j}$ represents the amount of good j produced by activity i

Two key assumptions restrict economy (A, B):

- **Assumption I:** (every good that is consumed is also produced)

$$b_{.,j} > \mathbf{0} \quad \forall j = 1, 2, \ldots, n$$

- **Assumption II:** (no free lunch)

$$a_{i,.} > \mathbf{0} \quad \forall i = 1, 2, \ldots, m$$

A semi-positive *intensity* m-vector x denotes levels at which activities are operated.

Therefore,

- vector $x^T A$ gives the total amount of *goods used in production*
- vector $x^T B$ gives *total outputs*

An economy (A, B) is said to be *productive*, if there exists a non-negative intensity vector $x \geq 0$ such that $x^T B > x^T A$.

The semi-positive n-vector p contains prices assigned to the n goods.

The p vector implies *cost* and *revenue* vectors

- the vector Ap tells *costs* of the vector of activities
- the vector Bp tells *revenues* from the vector of activities

Satisfaction or a property of an input-output pair (A, B) called *irreducibility* (or indecomposability) determines whether an economy can be decomposed into multiple "sub-economies".

Definition: For an economy (A, B), the set of goods $S \subset \{1, 2, \ldots, n\}$ is called an *independent subset* if it is possible to produce every good in S without consuming goods from outside S. Formally, the set S is independent if $\exists T \subset \{1, 2, \ldots, m\}$ (a subset of activities) such that $a_{i,j} = 0 \; \forall i \in T$ and $j \in S^c$ and for all $j \in S$, $\exists i \in T$ for which $b_{i,j} > 0$. The economy is **irreducible** if there are no proper independent subsets.

We study two examples, both in Chapter 9.6 of Gale [Gal89]

```
# (1) Irreducible (A, B) example: a_0 = β_0
A1 = np.array([[0, 1, 0, 0],
               [1, 0, 0, 1],
               [0, 0, 1, 0]])

B1 = np.array([[1, 0, 0, 0],
               [0, 0, 2, 0],
               [0, 1, 0, 1]])

# (2) Reducible (A, B) example: β_0 < a_0
A2 = np.array([[0, 1, 0, 0, 0, 0],
```

(continues on next page)

(continued from previous page)

```
                [1, 0, 1, 0, 0, 0],
                [0, 0, 0, 1, 0, 0],
                [0, 0, 1, 0, 0, 1],
                [0, 0, 0, 0, 1, 0]])

B2 = np.array([[1, 0, 0, 1, 0, 0],
                [0, 1, 0, 0, 0, 0],
                [0, 0, 1, 0, 0, 0],
                [0, 0, 0, 0, 2, 0],
                [0, 0, 0, 1, 0, 1]])
```

The following code sets up our first Neumann economy or `Neumann` instance

```
n1 = Neumann(A1, B1)
n1
```

```
Generalized von Neumann expanding model:
  - number of goods          : 4
  - number of activities     : 3

Assumptions:
  - AI:  every column of B has a positive entry    : True
  - AII: every row of A has a positive entry       : True
```

Here is a second instance of a Neumann economy

```
n2 = Neumann(A2, B2)
n2
```

```
Generalized von Neumann expanding model:
  - number of goods          : 6
  - number of activities     : 5

Assumptions:
  - AI:  every column of B has a positive entry    : True
  - AII: every row of A has a positive entry       : True
```

26.3 Dynamic Interpretation

Attach a time index t to the preceding objects, regard an economy as a dynamic system, and study sequences

$$\{(A_t, B_t)\}_{t\geq 0}, \qquad \{x_t\}_{t\geq 0}, \qquad \{p_t\}_{t\geq 0}$$

An interesting special case holds the technology process constant and investigates the dynamics of quantities and prices only.

Accordingly, in the rest of this lecture, we assume that $(A_t, B_t) = (A, B)$ for all $t \geq 0$.

A crucial element of the dynamic interpretation involves the timing of production.

We assume that production (consumption of inputs) takes place in period t, while the consequent output materializes in period $t + 1$, i.e., consumption of $x_t^T A$ in period t results in $x_t^T B$ amounts of output in period $t + 1$.

These timing conventions imply the following feasibility condition:

$$x_t^T B \geq x_{t+1}^T A \qquad \forall t \geq 1$$

which asserts that no more goods can be used today than were produced yesterday.

Accordingly, Ap_t tells the costs of production in period t and Bp_t tells revenues in period $t+1$.

26.3.1 Balanced Growth

We follow John von Neumann in studying "balanced growth".

Let $./$ denote an elementwise division of one vector by another and let $\alpha > 0$ be a scalar.

Then *balanced growth* is a situation in which

$$x_{t+1}./x_t = \alpha, \quad \forall t \geq 0$$

With balanced growth, the law of motion of x is evidently $x_{t+1} = \alpha x_t$ and so we can rewrite the feasibility constraint as

$$x_t^T B \geq \alpha x_t^T A \qquad \forall t$$

In the same spirit, define $\beta \in \mathbb{R}$ as the **interest factor** per unit of time.

We assume that it is always possible to earn a gross return equal to the constant interest factor β by investing "outside the model".

Under this assumption about outside investment opportunities, a no-arbitrage condition gives rise to the following (no profit) restriction on the price sequence:

$$\beta A p_t \geq B p_t \qquad \forall t$$

This says that production cannot yield a return greater than that offered by the outside investment opportunity (here we compare values in period $t+1$).

The balanced growth assumption allows us to drop time subscripts and conduct an analysis purely in terms of a time-invariant growth rate α and interest factor β.

26.4 Duality

Two problems are connected by a remarkable dual relationship between technological and valuation characteristics of the economy:

Definition: The *technological expansion problem* (TEP) for the economy (A, B) is to find a semi-positive m-vector $x > 0$ and a number $\alpha \in \mathbb{R}$ that satisfy

$$\begin{aligned} \max_{\alpha} \quad & \alpha \\ \text{s.t.} \quad & x^T B \geq \alpha x^T A \end{aligned}$$

Theorem 9.3 of David Gale's book [Gal89] asserts that if Assumptions I and II are both satisfied, then a maximum value of α exists and that it is positive.

The maximal value is called the *technological expansion rate* and is denoted by α_0. The associated intensity vector x_0 is the *optimal intensity vector*.

Definition: The economic expansion problem (EEP) for (A, B) is to find a semi-positive n-vector $p > 0$ and a number $\beta \in \mathbb{R}$ that satisfy

$$\begin{aligned} \min_{\beta} \quad & \beta \\ \text{s.t.} \quad & Bp \leq \beta A p \end{aligned}$$

Assumptions I and II imply existence of a minimum value $\beta_0 > 0$ called the *economic expansion rate*.

The corresponding price vector p_0 is the *optimal price vector*.

Because the criterion functions in the *technological expansion* problem and the *economical expansion problem* are both linearly homogeneous, the optimality of x_0 and p_0 are defined only up to a positive scale factor.

For convenience (and to emphasize a close connection to zero-sum games), we normalize both vectors x_0 and p_0 to have unit length.

A standard duality argument (see Lemma 9.4. in (Gale, 1960) [Gal89]) implies that under Assumptions I and II, $\beta_0 \leq \alpha_0$.

But to deduce that $\beta_0 \geq \alpha_0$, Assumptions I and II are not sufficient.

Therefore, von Neumann [vN37] went on to prove the following remarkable "duality" result that connects TEP and EEP.

Theorem 1 (von Neumann): If the economy (A, B) satisfies Assumptions I and II, then there exist (γ^*, x_0, p_0), where $\gamma^* \in [\beta_0, \alpha_0] \subset \mathbb{R}$, $x_0 > 0$ is an m-vector, $p_0 > 0$ is an n-vector, and the following arbitrage true

$$
\begin{aligned}
x_0^T B &\geq \gamma^* x_0^T A \\
B p_0 &\leq \gamma^* A p_0 \\
x_0^T \left(B - \gamma^* A \right) p_0 &= 0
\end{aligned}
$$

Note: *Proof (Sketch):* Assumption I and II imply that there exist (α_0, x_0) and (β_0, p_0) that solve the TEP and EEP, respectively. If $\gamma^* > \alpha_0$, then by definition of α_0, there cannot exist a semi-positive x that satisfies $x^T B \geq \gamma^* x^T A$. Similarly, if $\gamma^* < \beta_0$, there is no semi-positive p for which $Bp \leq \gamma^* Ap$. Let $\gamma^* \in [\beta_0, \alpha_0]$, then $x_0^T B \geq \alpha_0 x_0^T A \geq \gamma^* x_0^T A$. Moreover, $B p_0 \leq \beta_0 A p_0 \leq \gamma^* A p_0$. These two inequalities imply $x_0 \left(B - \gamma^* A \right) p_0 = 0$.

Here the constant γ^* is both an expansion factor and an interest factor (not necessarily optimal).

We have already encountered and discussed the first two inequalities that represent feasibility and no-profit conditions.

Moreover, the equality $x_0^T \left(B - \gamma^* A \right) p_0 = 0$ concisely expresses the requirements that if any good grows at a rate larger than γ^* (i.e., if it is *oversupplied*), then its price must be zero; and that if any activity provides negative profit, it must be unused.

Therefore, the conditions stated in Theorem I ex encode all equilibrium conditions.

So Theorem I essentially states that under Assumptions I and II there always exists an equilibrium (γ^*, x_0, p_0) with balanced growth.

Note that Theorem I is silent about uniqueness of the equilibrium. In fact, it does not rule out (trivial) cases with $x_0^T B p_0 = 0$ so that nothing of value is produced.

To exclude such uninteresting cases, Kemeny, Morgenstern and Thomspson [KMT56] add an extra requirement

$$
x_0^T B p_0 > 0
$$

and call the associated equilibria *economic solutions*.

They show that this extra condition does not affect the existence result, while it significantly reduces the number of (relevant) solutions.

26.5 Interpretation as Two-player Zero-sum Game

To compute the equilibrium (γ^*, x_0, p_0), we follow the algorithm proposed by Hamburger, Thompson and Weil (1967), building on the key insight that an equilibrium (with balanced growth) can be solves a particular two-player zero-sum game. First, we introduce some notation.

Consider the $m \times n$ matrix C as a payoff matrix, with the entries representing payoffs from the **minimizing** column player to the **maximizing** row player and assume that the players can use mixed strategies. Thus,

- the row player chooses the m-vector $x > \mathbf{0}$ subject to $\iota_m^T x = 1$
- the column player chooses the n-vector $p > \mathbf{0}$ subject to $\iota_n^T p = 1$.

Definition: The $m \times n$ matrix game C has the *solution* $(x^*, p^*, V(C))$ in mixed strategies if

$$(x^*)^T C e^j \geq V(C) \quad \forall j \in \{1, \dots, n\} \qquad \text{and} \qquad (e^i)^T C p^* \leq V(C) \quad \forall i \in \{1, \dots, m\}$$

The number $V(C)$ is called the *value* of the game.

From the above definition, it is clear that the value $V(C)$ has two alternative interpretations:

- by playing the appropriate mixed stategy, the maximizing player can assure himself at least $V(C)$ (no matter what the column player chooses)
- by playing the appropriate mixed stategy, the minimizing player can make sure that the maximizing player will not get more than $V(C)$ (irrespective of what is the maximizing player's choice)

A famous theorem of Nash (1951) tells us that there always exists a mixed strategy Nash equilibrium for any *finite* two-player zero-sum game.

Moreover, von Neumann's Minmax Theorem [vN28] implies that

$$V(C) = \max_x \min_p \; x^T C p = \min_p \max_x \; x^T C p = (x^*)^T C p^*$$

26.5.1 Connection with Linear Programming (LP)

Nash equilibria of a finite two-player zero-sum game solve a linear programming problem.

To see this, we introduce the following notation

- For a fixed x, let v be the value of the minimization problem: $v \equiv \min_p x^T C p = \min_j x^T C e^j$
- For a fixed p, let u be the value of the maximization problem: $u \equiv \max_x x^T C p = \max_i (e^i)^T C p$

Then the *max-min problem* (the game from the maximizing player's point of view) can be written as the *primal* LP

$$\begin{aligned} V(C) = \max \quad & v \\ \text{s.t.} \quad & v\iota_n^T \leq x^T C \\ & x \geq \mathbf{0} \\ & \iota_n^T x = 1 \end{aligned}$$

while the *min-max problem* (the game from the minimizing player's point of view) is the *dual* LP

$$\begin{aligned} V(C) = \min \quad & u \\ \text{s.t.} \quad & u\iota_m \geq Cp \\ & p \geq \mathbf{0} \\ & \iota_m^T p = 1 \end{aligned}$$

Hamburger, Thompson and Weil [HTW67] view the input-output pair of the economy as payoff matrices of two-player zero-sum games.

Using this interpretation, they restate Assumption I and II as follows

$$V(-A) < 0 \qquad \text{and} \qquad V(B) > 0$$

Note: *Proof (Sketch)*:

- $\Rightarrow V(B) > 0$ implies $x_0^T B \gg \mathbf{0}$, where x_0 is a maximizing vector. Since B is non-negative, this requires that each column of B has at least one positive entry, which is Assumption I.
- $\Leftarrow$ From Assumption I and the fact that $p > \mathbf{0}$, it follows that $Bp > \mathbf{0}$. This implies that the maximizing player can always choose x so that $x^T Bp > 0$ so that it must be the case that $V(B) > 0$.

In order to (re)state Theorem I in terms of a particular two-player zero-sum game, we define a matrix for $\gamma \in \mathbb{R}$

$$M(\gamma) \equiv B - \gamma A$$

For fixed γ, treating $M(\gamma)$ as a matrix game, calculating the solution of the game implies

- If $\gamma > \alpha_0$, then for all $x > 0$, there $\exists j \in \{1, \dots, n\}$, s.t. $[x^T M(\gamma)]_j < 0$ implying that $V(M(\gamma)) < 0$.
- If $\gamma < \beta_0$, then for all $p > 0$, there $\exists i \in \{1, \dots, m\}$, s.t. $[M(\gamma)p]_i > 0$ implying that $V(M(\gamma)) > 0$.
- If $\gamma \in \{\beta_0, \alpha_0\}$, then (by Theorem I) the optimal intensity and price vectors x_0 and p_0 satisfy

$$x_0^T M(\gamma) \geq \mathbf{0}^T \qquad \text{and} \qquad M(\gamma)p_0 \leq \mathbf{0}$$

That is, $(x_0, p_0, 0)$ is a solution of the game $M(\gamma)$ so that $V\left(M(\beta_0)\right) = V\left(M(\alpha_0)\right) = 0$.

- If $\beta_0 < \alpha_0$ and $\gamma \in (\beta_0, \alpha_0)$, then $V(M(\gamma)) = 0$.

Moreover, if x' is optimal for the maximizing player in $M(\gamma')$ for $\gamma' \in (\beta_0, \alpha_0)$ and p'' is optimal for the minimizing player in $M(\gamma'')$ where $\gamma'' \in (\beta_0, \gamma')$, then $(x', p'', 0)$ is a solution for $M(\gamma)$ $\forall \gamma \in (\gamma'', \gamma')$.

Proof (Sketch): If x' is optimal for a maximizing player in game $M(\gamma')$, then $(x')^T M(\gamma') \geq \mathbf{0}^T$ and so for all $\gamma < \gamma'$.

$$(x')^T M(\gamma) = (x')^T M(\gamma') + (x')^T(\gamma' - \gamma)A \geq \mathbf{0}^T$$

hence $V(M(\gamma)) \geq 0$. If p'' is optimal for a minimizing player in game $M(\gamma'')$, then $M(\gamma)p \leq \mathbf{0}$ and so for all $\gamma'' < \gamma$

$$M(\gamma)p'' = M(\gamma'') + (\gamma'' - \gamma)Ap'' \leq \mathbf{0}$$

hence $V(M(\gamma)) \leq 0$.

It is clear from the above argument that β_0, α_0 are the minimal and maximal γ for which $V(M(\gamma)) = 0$.

Furthermore, Hamburger et al. [HTW67] show that the function $\gamma \mapsto V(M(\gamma))$ is continuous and nonincreasing in γ.

This suggests an algorithm to compute (α_0, x_0) and (β_0, p_0) for a given input-output pair (A, B).

26.5.2 Algorithm

Hamburger, Thompson and Weil [HTW67] propose a simple bisection algorithm to find the minimal and maximal roots (i.e. β_0 and α_0) of the function $\gamma \mapsto V(M(\gamma))$.

Step 1

First, notice that we can easily find trivial upper and lower bounds for α_0 and β_0.

- TEP requires that $x^T(B - \alpha A) \geq \mathbf{0}^T$ and $x > \mathbf{0}$, so if α is so large that $\max_i\{[(B - \alpha A)\iota_n]_i\} < 0$, then TEP ceases to have a solution.

Accordingly, let **UB** be the α^* that solves $\max_i\{[(B - \alpha^* A)\iota_n]_i\} = 0$.

- Similar to the upper bound, if β is so low that $\min_j\{[\iota_m^T(B - \beta A)]_j\} > 0$, then the EEP has no solution and so we can define **LB** as the β^* that solves $\min_j\{[\iota_m^T(B - \beta^* A)]_j\} = 0$.

The *bounds* method calculates these trivial bounds for us

```
n1.bounds()
```

```
(1.0, 2.0)
```

Step 2

Compute α_0 and β_0

- Finding α_0
  1. Fix $\gamma = \frac{UB+LB}{2}$ and compute the solution of the two-player zero-sum game associated with $M(\gamma)$. We can use either the primal or the dual LP problem.
  2. If $V(M(\gamma)) \geq 0$, then set $LB = \gamma$, otherwise let $UB = \gamma$.
  3. Iterate on 1. and 2. until $|UB - LB| < \epsilon$.
- Finding β_0
  1. Fix $\gamma = \frac{UB+LB}{2}$ and compute the solution of the two-player zero-sum game associated. with $M(\gamma)$. We can use either the primal or the dual LP problem.
  2. If $V(M(\gamma)) > 0$, then set $LB = \gamma$, otherwise let $UB = \gamma$.
  3. Iterate on 1. and 2. until $|UB - LB| < \epsilon$.
- *Existence*: Since $V(M(LB)) > 0$ and $V(M(UB)) < 0$ and $V(M(\cdot))$ is a continuous, nonincreasing function, there is at least one $\gamma \in [LB, UB]$, s.t. $V(M(\gamma)) = 0$.

The *zerosum* method calculates the value and optimal strategies associated with a given γ.

```
y = 2

print(f'Value of the game with y = {y}')
print(n1.zerosum(y=y)[0])
print('Intensity vector (from the primal)')
print(n1.zerosum(y=y)[1])
print('Price vector (from the dual)')
print(n1.zerosum(y=y, dual=True)[1])
```

```
Value of the game with y = 2
-0.24000000097850305
Intensity vector (from the primal)
[0.32 0.28 0.4 ]
Price vector (from the dual)
[4.00e-01 3.20e-01 2.80e-01 2.54e-10]
```

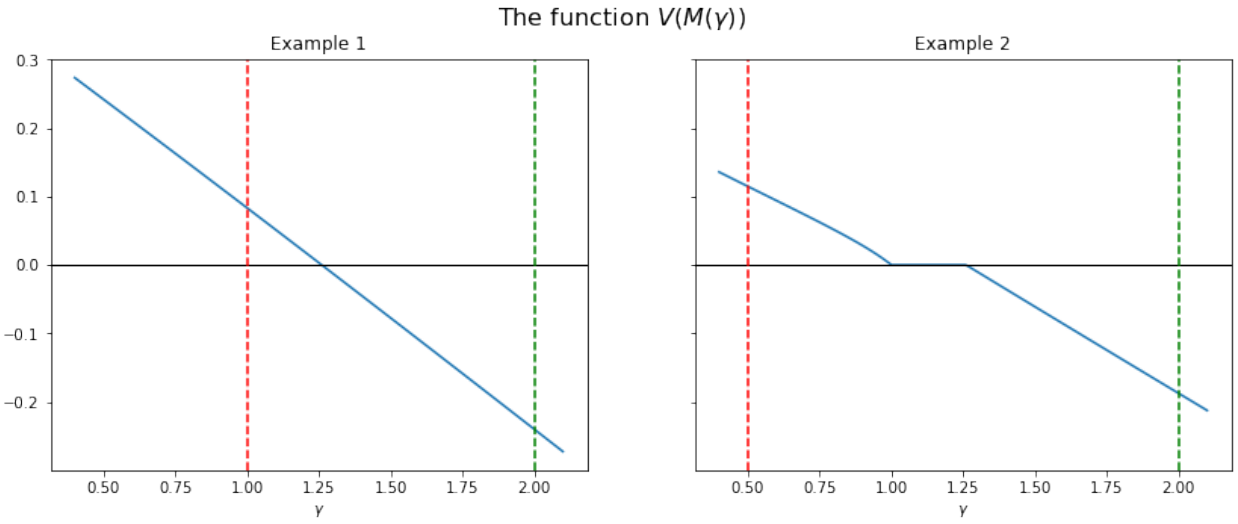
```
numb_grid = 100
y_grid = np.linspace(0.4, 2.1, numb_grid)

value_ex1_grid = np.asarray([n1.zerosum(y=y_grid[i])[0]
                             for i in range(numb_grid)])
value_ex2_grid = np.asarray([n2.zerosum(y=y_grid[i])[0]
                             for i in range(numb_grid)])

fig, axes = plt.subplots(1, 2, figsize=(14, 5), sharey=True)
fig.suptitle(r'The function $V(M(\gamma))$', fontsize=16)

for ax, grid, N, i in zip(axes, (value_ex1_grid, value_ex2_grid),
                          (n1, n2), (1, 2)):
    ax.plot(y_grid, grid)
    ax.set(title=f'Example {i}', xlabel='$\gamma$')
    ax.axhline(0, c='k', lw=1)
    ax.axvline(N.bounds()[0], c='r', ls='--', label='lower bound')
    ax.axvline(N.bounds()[1], c='g', ls='--', label='upper bound')

plt.show()
```



The *expansion* method implements the bisection algorithm for α_0 (and uses the primal LP problem for x_0)

```
a_0, x, p = n1.expansion()
print(f'a_0 = {a_0}')
print(f'x_0 = {x}')
print(f'The corresponding p from the dual = {p}')
```

```
a_0 = 1.2599210478365421
x_0 = [0.33 0.26 0.41]
The corresponding p from the dual = [4.13e-01 3.27e-01 2.60e-01 1.82e-10]
```

The *interest* method implements the bisection algorithm for β_0 (and uses the dual LP problem for p_0)

```
β_0, x, p = n1.interest()
print(f'β_0 = {β_0}')
print(f'p_0 = {p}')
print(f'The corresponding x from the primal = {x}')
```

```
β_0 = 1.2599210478365421
p_0 = [4.13e-01 3.27e-01 2.60e-01 1.82e-10]
The corresponding x from the primal = [0.33 0.26 0.41]
```

Of course, when γ^* is unique, it is irrelevant which one of the two methods we use – both work.

In particular, as will be shown below, in case of an irreducible (A, B) (like in Example 1), the maximal and minimal roots of $V(M(\gamma))$ necessarily coincide implying a "full duality" result, i.e. $\alpha_0 = \beta_0 = \gamma^*$ so that the expansion (and interest) rate γ^* is unique.

26.5.3 Uniqueness and Irreducibility

As an illustration, compute first the maximal and minimal roots of $V(M(\cdot))$ for our Example 2 that has a reducible input-output pair (A, B)

```
α_0, x, p = n2.expansion()
print(f'α_0 = {α_0}')
print(f'x_0 = {x}')
print(f'The corresponding p from the dual = {p}')
```

```
α_0 = 1.1343231229111552
x_0 = [1.67e-11 1.83e-11 3.24e-01 2.61e-01 4.15e-01]
The corresponding p from the dual = [5.04e-01 4.96e-01 2.96e-12 2.24e-12 3.08e-12 3.
↪56e-12]
```

```
β_0, x, p = n2.interest()
print(f'β_0 = {β_0}')
print(f'p_0 = {p}')
print(f'The corresponding x from the primal = {x}')
```

```
β_0 = 1.2579826870933175
p_0 = [5.11e-01 4.89e-01 2.73e-08 2.17e-08 1.88e-08 2.66e-09]
The corresponding x from the primal = [1.61e-09 1.65e-09 3.27e-01 2.60e-01 4.12e-01]
```

As we can see, with a reducible (A, B), the roots found by the bisection algorithms might differ, so there might be multiple γ^* that make the value of the game with $M(\gamma^*)$ zero. (see the figure above).

Indeed, although the von Neumann theorem assures existence of the equilibrium, Assumptions I and II are not sufficient for uniqueness. Nonetheless, Kemeny et al. (1967) show that there are at most finitely many economic solutions, meaning that there are only finitely many γ^* that satisfy $V(M(\gamma^*)) = 0$ and $x_0^T B p_0 > 0$ and that for each such γ_i^*, there is a self-contained part of the economy (a sub-economy) that in equilibrium can expand independently with the expansion coefficient γ_i^*.

The following theorem (see Theorem 9.10. in Gale [Gal89]) asserts that imposing irreducibility is sufficient for uniqueness of (γ^*, x_0, p_0).

Theorem II: Adopt the conditions of Theorem 1. If the economy (A, B) is irreducible, then $\gamma^* = \alpha_0 = \beta_0$.

26.5.4 A Special Case

There is a special (A, B) that allows us to simplify the solution method significantly by invoking the powerful Perron-Frobenius theorem for non-negative matrices.

Definition: We call an economy *simple* if it satisfies

- $n = m$
- Each activity produces exactly one good
- Each good is produced by one and only one activity.

These assumptions imply that $B = I_n$, i.e., that B can be written as an identity matrix (possibly after reshuffling its rows and columns).

The simple model has the following special property (Theorem 9.11. in Gale [Gal89]): if x_0 and $\alpha_0 > 0$ solve the TEP with (A, I_n), then

$$x_0^T = \alpha_0 x_0^T A \qquad \Leftrightarrow \qquad x_0^T A = \left(\frac{1}{\alpha_0}\right) x_0^T$$

The latter shows that $1/\alpha_0$ is a positive eigenvalue of A and x_0 is the corresponding non-negative left eigenvector.

The classic result of **Perron and Frobenius** implies that a non-negative matrix has a non-negative eigenvalue-eigenvector pair.

Moreover, if A is irreducible, then the optimal intensity vector x_0 is positive and *unique* up to multiplication by a positive scalar.

Suppose that A is reducible with k irreducible subsets $S_1, \ldots, S_k$. Let A_i be the submatrix corresponding to S_i and let α_i and β_i be the associated expansion and interest factors, respectively. Then we have

$$\alpha_0 = \max_i \{\alpha_i\} \qquad \text{and} \qquad \beta_0 = \min_i \{\beta_i\}$$