To appear in
SPRINGER NATURE OPERATIONS RESEARCH FORUM

Model Development with Maple in PhD-level Management Science Courses: A Personal Account[1,2]

Mahmut Parlar

DeGroote School of Business

McMaster University

Hamilton, Ontario L8S 4M4

Canada

July 2021 / Revised October 2021 and February 2022

[1]Research supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

[2]The author is grateful to Dr. J. D. Pintér and two referees for their constructive comments.

Contents

Abstract

I have used the computer algebra system Maple for more than 30 years in my research and three PhD-level management science courses I have taught. I also wrote a book on Maple to illustrate its successful use in solving operations research / management science problems. In this paper I will first present a detailed presentation of Maple's use in my course on Stochastic Processes and include relevant Maple worksheets. This is followed by a description of Maple's use in my course on Dynamic Programming and Optimal Control. The paper ends with a discussion of my third PhD course on Game Theory. The complete list of all Maple worksheets presented are available on my homepage at `https://profs.degroote.mcmaster.ca/ads/parlar/ORMapleBook/Parlar-Supplements-SpringerORForum.zip`.

1 Introduction

Computer Algebra Systems (CASs) are sophisticated mathematical software that can perform symbolic manipulations, perform complicated numerical computations and generate publication-quality graphs. For example, they can symbolically integrate and differentiate functions and solve differential equations. They can also integrate functions numerically for which there exist no closed-form results [such as the integral of $\exp(-x^2)$], and numerically solve ordinary and partial differential equations (and their systems). Many of these CASs are also capable of plotting 2-D and 3-D graphs of functions, plot implicit functions and contours of 3-D functions, among others.

Many CASs have a very rich knowledge base and they can perform impressive mathematical feats such as solving linear and non-linear optimization problems, manipulating Laplace transforms and their inverses, performing symbolic random variable calculations, and performing symbolic and numerical calculations in linear algebra. Depending on the availability of the specific packages, they can also perform financial analyses such as calculating the price of American and European options, curve fitting and mathematical logic. Some CASs can also solve general *global* optimization problems; see, Pintér ([29] and [30]). I mention Pintér's books here as his Lipschitz Global Optimizer (LGO) was the engine behind the first Global Optimization Toolbox (GOT) adopted by Maplesoft in Maple 9.5.

The knowledge base of the most advanced CASs cover nearly 3,000 years of accumulation of mathematical knowledge, from basic Euclidean geometry to the currently popular deep learning with neural networks. In addition to their mathematical "expertise," many CASs can export their input/output to LaTeX, and translate their native code to Fortran, C and Python, among others. They can also import/export data sets to and from other statistical software, such as R, Excel and SAS. The impressive abilities of the CASs listed above have made them a clear choice for many instructors teaching advanced technical courses in engineering and science faculties, and PhD-level operations research/management science (OR/MS) programs in business schools.

The earliest fully-functional CAS was Macsyma ("Project MAC's SYmbolic MAnipulator") which was originally developed in 1968 at MIT[1]. I never had the opportunity to experiment with Macsyma, but I had read about it in an advertisement piece in the September 1984 issue of Scientific American. The Macsyma example shown was the computation of an indefinite integral involving the error function $\text{erf}(x) = \int_0^x (2/\sqrt{\pi}) e^{-u^2}\, du$. The closed-form

[1] `https://en.wikipedia.org/wiki/Macsyma`

solution for the integral was found as,

$$\int \operatorname{erf}(ax)\operatorname{erf}(bx)\ dx = -\frac{\sqrt{a^2+b^2}\operatorname{erf}(x+\sqrt{a^2+b^2})}{ab\sqrt{\pi}} + x\operatorname{erf}(ax)\operatorname{erf}(bx) + \frac{e^{-a^2x^2}\operatorname{erf}(bx)}{a\sqrt{\pi}} + \frac{e^{-b^2x^2}\operatorname{erf}(ax)}{b\sqrt{\pi}},$$

where a and b are **any** real numbers. The advertisement stated that Macsyma can solve equations, differentiate functions, compute Laplace transforms and manipulate matrices and it would operate on mainframe and minicomputers. In the early 1970s I had taken courses in computer programming and numerical analysis with Fortran IV and I had used the IBM 360 mainframe computer, thus I was familiar with scientific computation. But the fact that computers could now perform symbolic mathematics fascinated me. I should note that Maple can also evaluate this integral easily as I have shown in ▶Macsyma-Integral-1.mw◀. This and every other Maple file I refer to in this paper are available on my homepage at

https://profs.degroote.mcmaster.ca/ads/parlar/ORMapleBook/Parlar-Supplements-SpringerORForum.zip.

I had my first hands-on experience with a CAS in the early 1980s when I purchased the muMATH[2] system which ran on my IBM-PC. (I had seen brief reviews of this software by Edwards [6] and Williams [36] in BYTE magazine.) The syntax was simple and somewhat crude, but it worked. muMATH was able to solve equations symbolically, perform symbolic integration and it could even solve ordinary differential equations. All commands had to be in capitals and the result was also printed in capitals. For example, to evaluate $\int_a^b x^2\ dx$, one would enter DEFINT(X^2, X, A, B) and the result would appear as (-A^3 + B^3)/3, all in 1-D math notation. A more user-friendly version of muMATH appeared in late 1980s under the new name, Derive[3] which could do more and generate good-quality graphs in multiple windows.

Currently, there are several dozen CASs available in the market[4] which can run under different operating systems including Windows, macOS, Linux and others. Among these CASs, the market leaders appear to be Maple[5] from Waterloo Maple, and Wolfram Mathematica[6] from Wolfram Research. My first encounter with Maple was in the late 1980s when McMaster University had a licence to run Maple on a DEC VAX minicomputer. I continued

[2] https://en.wikipedia.org/wiki/MuMATH
[3] https://en.wikipedia.org/wiki/Derive_(computer_algebra_system)
[4] https://en.wikipedia.org/wiki/List_of_computer_algebra_systems
[5] https://en.wikipedia.org/wiki/Maple_(software)
[6] https://en.wikipedia.org/wiki/Wolfram_Mathematica

using Maple on VAX until I purchased the Windows version of Maple V R4 ("V" meaning "Visual") which was released in January 1996. The graphical user interface feature ("V") of this version made it so much easier to use and provided several new other mathematical functions.

When version R5.1 was released in 1998 with even more features, I decided to use this powerful software to write an operations research book which was published in 2000 (Parlar [26]). With the experience I gained while writing my book, I became convinced that Maple can be an excellent pedagogical tool in the three PhD-level management science courses I teach in DeGroote School of Business at McMaster University, i.e., (i) Stochastic Processes with Business Applications, (ii) Dynamic Programming and Optimal Control, and (iii) Game Theory and Decision Analysis. As I will demonstrate in subsequent sections, Maple was very useful in solving many problems analytically (i.e., in closed-form) which were not previously possible. I will also describe several problems which can be solved numerically with Maple using only a few lines of code, rather than a complicated set of commands using C++, Fortran, etc. The examples presented in [26] have constituted the building blocks of the courses mentioned above where Maple was used extensively in model building.

Since the late 1980s I have been teaching our PhD-level stochastic processes course. In Section 2, I will describe how I have used Maple in this course since the early 2000s to illustrate concepts in stochastic processes and solve realistic problems. My research interests in dynamic optimization led me to teach a second PhD-level course on dynamic programming and optimal control. In Section 3, I will show how I used Maple in this course where Maple's symbolic manipulation capabilities allowed me to present difficult and realistic problems. I have also been interested in game theory applications in supply chain management and have published several papers on this topic. About 10 years ago I decided to leverage my interest and experience on this topic and offered a PhD-level course on game theory and decision analysis. In Section 4, I will show how Maple was again helpful in illustrating difficult concepts and help solve realistic problems. I will conclude the paper in Section 5 with a few comments on other potential uses of Maple in advanced operations research/management science courses. With certain exceptions, I will not include the Maple output of the examples as they could be quite lengthy. Instead, for each example, I will make available for the reader two files; one with the extension `.mw` (Maple Worksheet); the other, the `.pdf` export of the `.mw` file. As mentioned above, these files will be available on my homepage at `https://profs.degroote.mcmaster.ca/ads/parlar/ORMapleBook/Parlar-Supplements-SpringerORForum.zip`.

2 Stochastic Processes with Business Applications (PhD course)

I have been teaching our PhD course Bus Q771: Stochastic Processes with Business Applications since the late 1980s. In my lectures I make use of Kao [12] and Ross [31]. The former has excellent examples of the theoretical material covered in the latter. I also refer to Parlar [26] for the Maple-related materials.

2.1 Review of Basic Probability Theory

After defining discrete- and continuos-time stochastic processes, I give some examples of such processes and then move on to a quick review of probability theory. I give the definition of a probability measure on σ-fields and continue with random variables and their moments. One example I discuss involves two independent uniform random variables defined on (a, b), i.e., $X \sim U(a, b)$ and $Y \sim U(a, b)$. When $(a, b) = (0, 1)$ it is easy to find the p.d.f. of the sum $X + Y$ (which is triangular) and also evaluate probabilities such as $\Pr(X \leq Y) = \frac{1}{2}$. When (a, b) are not specified, the p.d.f. of the sum $f_{X+Y}(t)$ is somewhat challenging to determine. The Maple file ►Two-Uniforms.mw◄ does this nicely for general (a, b) and produces the following result for the density of the sum $X + Y$:

$$f_{X+Y}(t) = \begin{cases} 0, & \text{if } t > a + b \text{ and } t > 2b, \\ \dfrac{2b - t}{(b - a)^2}, & \text{if } t > a + b \text{ and } t < 2b, \\ \dfrac{-2a + t}{(b - a)^2}, & \text{if } t < a + b \text{ and } t > 2a, \\ 0, & \text{if } t < a + b \text{ and } t < 2a. \end{cases}$$

If a third uniform is involved as $Z \sim U(a, b)$, the p.d.f. of the sum $f_{X+Y+Z}(t)$ can still be obtained relatively easily with Maple as shown in ►Three-Uniforms.mw◄.

When I discuss jointly-distributed random variables, I give an example of the bivariate normal and plot the surface of the joint p.d.f. in ►Bivariate-Normal-Plot.mw◄.

Probability generating functions (p.g.f.) and their inverses play an important role in applied probability. Consider a discrete random variable X with the probability density function $a(k) = \Pr(X = k)$, $k = 0, 1, \ldots$ and $\sum_{k=0}^{\infty} a(k) = 1$. The p.g.f. of X is defined as $\Pi_X(z) = E(z^X) = \sum_{k=0}^{\infty} a(k) z^k$ from which we obtain $E(X) = \Pi'_X(1)$ as the expected value and $\mathrm{Var}(X) = \Pi''_X(1) - [\Pi'_X(1)]^2 + \Pi'_X(1)$ as variance of X. If $\Pi_X(z)$ is a general function of

z, then $a(k)$ are obtained as,

$$a(k) = \frac{1}{k!}\left(\frac{d^k \Pi_X(z)}{dz^k}\right)_{z=0}, \quad k = 0, 1, \ldots,$$

see, Kao [12, Ch. 1] and Medhi [20, Ch. 1]. In the example I use, we have,

$$\Pi(z) = \frac{4}{(2-z)(3-z)^2}.$$

After loading the `with(genfunc)` package, we use `a := unapply(rgf_expand(PI, z, n), n)` to invert the p.g.f. Maple obtains the explicit result in ▶`Probability-Generating-Function.mw`◀.

For continuous random variables (r.v.), Laplace transforms (LT) and their inverses play an important role in applied probability. For a nonnegative r.v. X with p.d.f. $f_X(t)$, its Laplace transform is defined $\tilde{f}(s) = \int_0^\infty e^{-st} f_X(t)\, dt$. A large number of these transforms can be inverted analytically by referring to the widely available tables. In ▶`Laplace-Transform.mw`◀, I start with the LT of the Erlang$(4, \lambda)$ r.v. as $\tilde{f}(s) = \lambda^4/(s+\lambda)^4$ and invert it to find this Erlang's density as $f(t) = \frac{1}{6}\lambda^4 t^3 e^{-\lambda t}$. In another example, I consider a LT given as $\tilde{g}(s) = 1/((1+\sqrt{s})(s^2+\sqrt{s}))$ for which (to my knowledge) no closed-form inverse exists. Since the 2020 version, Maple has become capable of numerically inverting LTs based on Abate and Whitt's paper [1]. I use this feature of Maple and find the numerical inverse of $\tilde{g}(s)$ as $g(t)$ and plot it in ▶`Laplace-Transform.mw`◀.

2.2 Exponential Distribution and the Poisson Process

I start this topic by showing that the exponential r.v. X with rate λ, mean $E(X) = 1/\lambda$, and p.d.f. $f(x) = \lambda e^{-\lambda x}$ is memoryless. (In fact, it is the only memoryless continuous r.v.) I then connect it to the Poisson process by stating that for this process with rate λ, the interarrival times are always exponential with the same rate. To simulate the Poisson process with exponential interarrival times, I load the `with(Statistics)` package and define the r.v. as `X := RandomVariable(Exponential(1/lambda))` where $\lambda = 0.1$. I generate a sample of 20 variates and use them to generate the Poisson arrivals with a plot. This material is presented in the file ▶`Poisson-Simulation.mw`◀.

The above example is about simulating the Poisson process $\{N(t),\ t \geq 0\}$ as a special type of counting process which requires the generation of exponential random variates as the interarrival times. Maple can also generate random variates from the Poisson random variable N with probability mass function $\Pr(N = n) = e^{-\lambda}\lambda^n/n!$. I do this in the file

▶Poisson-Sampling.mw◀ with $\lambda = 5$ where 100 samples are generated. First, I define `N := RandomVariable(Poisson(5))` and then enter `A := Sample(N, 10^2)` to generate the 100 samples. The histogram of the sampled values are plotted and compared against the theoretical histogram.

When the arrival rate $\lambda(t)$ of a Poisson process becomes time-dependent (as in arrivals to a restaurant), the problem becomes more complicated. In ▶M(t)-M-1-1.mw◀, I consider a queueing system with a non-homogeneous Poisson arrival process, exponential service times, one server and a system capacity of one space. The arrival rate is $\lambda(t) = 100t(1-t)$ and the service rate is $\mu = 7$. For this problem we compute the transient probabilities for $t \in [0, 1]$ by solving a system of two DEs with variable coefficients. I also plot the $p_1(t)$ function corresponding to the probability of a full system at time t.

2.3 Renewal Theory

Renewal theory extends the exponential/Poisson duality with the more general assumption that the interarrival times of a counting process can be any nonnegative i.i.d. random variables. In the simpler Poisson case the expected number of occurrences in the interval $(0, t]$ is simply $M(t) = E[N(t)] = \lambda t$. In the renewal theory framework, this expected number (also known as the "renewal function") is generalized to $M(t) = \int_0^t \lambda(u)\, du$ where $\lambda(u)$ is the arrival rate at t. Calculation of the renewal function can be challenging if the interarrival times have complicated densities.

If a Laplace transform can be developed for the renewal function, Maple can successfully invert it (numerically, if necessary). A related quantity is the renewal "density" which is defined as the derivative of the renewal function, i.e., $m(t) = M'(t)$. It can be shown that the LT $\tilde{m}(s)$ of $m(t)$ can be expressed in terms of the LT $\tilde{f}(s)$ of the interarrival time density $f(t)$ as $\tilde{m}(s) = \tilde{f}(s)/[1 - \tilde{f}(s)]$. If $\tilde{m}(s)$ can be inverted to obtain, one can then find the renewal function from $M(t) = \int_0^t m(u)\, du$ as a result of the Fundamental Theorem of Calculus [noting that $M(0) = 0$]. In ▶Erlang21.mw◀, I assume that $f(t)$ is Erlang(n, λ) with parameters $(n, \lambda) = (2, 1)$ and with density $f(t) = te^{-t}$ for which the LT is given as $\tilde{f}(s) = 1/(1+s)^2$. Inverting $\tilde{m}(s)$ we find $M(t) = -1/4 + t/2 + e^{-2t}/4$. A plot of $M(t)$ with its asymptote and the linear approximation is also provided.

Why does Poisson naturally occur in practice? The answer is if there is a very large number of potential "customers" each with a very long mean interarrival time, one can use renewal theory to prove that the interarrivals of the superposed process is exponential. I prove this in the course using the forward recurrence time concept. This may seem surprising, so in ▶Poisson-Natural-Uniform.mw◀, I assume that there are 20 independent

renewal processes (which is large enough for our purposes) each with uniformly-distributed interarrival times with mean 500. Using simulation, I show that the superposed process is approximately Poisson. I also present a histogram of the simulated interarrival times of the superposed process.

Renewal Reward Theorem (RRT) is a powerful tool in formulating optimization models of regenerative stochastic processes, such as those encountered in inventory theory and queueing. Once a regenerative cycle is identified, one calculates the average cost per time AC simply as the ratio of the expected cycle cost EC to the expected cycle length EL. The average cost expression is normally a nonlinear function of one or more decision variables which can be optimized using standard tools of nonlinear programming. In ▶`Car-Buying.mw`◀, I consider a car replacement problem where the lifetime of a car X is random with c.d.f. $F(x)$. If the car breaks down before the end of its life, the owner must incur a breakdown cost of c_1 and the new car purchase cost of c_2. Using standard conditioning arguments, it is shown that $EC(T) = c_1 + c_2 F(T)$ and $EL(T) = T[1 - F(T)] + \int_0^T x\, dF(x)$ where T is the time of replacement. Assuming $X \sim U[0, 10]$ and $(c_1, c_2) = (3, \frac{1}{2})$ the average cost is found as $AC(T) = g(T) = (60+T)/(20T-T^2)$. Using `with(Optimization)` and `NLPSolve()`, the optimal replacement time is found as $T^* = 9.28$ and $g(T^*) = 0.69$. I also provide a graph of the $g(T)$ function over $[0, 20]$.

2.4 Discrete-time Markov Chains

I start this chapter by considering a discrete-time stochastic process $\{X_n,\, n = 0, 1, 2, \dots\}$ with a finite or infinite state space $\mathcal{S}$ and the property that for any set of values $j, i, i_{n-1}, \dots, i_0$ belonging to $\mathcal{S}$,

$$\Pr(X_{n+1} = j \mid X_n = i,\, X_{n-1} = i_{n-1}, \dots,\, X_0 = i_0) = \Pr(X_{n+1} = j \mid X_n = i) = p_{ij}^{(n,n+1)},$$

for $n = 0, 1, 2, \dots$. A *time-homogeneous* discrete-time Markov chain (DTMC) has the property that the transition probabilities $p_{ij}^{(n,n+1)}$ are constant over time; i.e., for all $n = 0, 1, 2, \dots$, we have $\Pr(X_{n+1} = j \mid X_n = i) = p_{ij},\ i, j \in \mathcal{S}$.

The first example I discuss involves a periodic-review inventory system with random demand. Under an (s, S) ordering policy, I show that the inventory level process X_n is a DTMC since X_{n+1} depends on X_n, and some parameters. For the special case of Poisson demand in each period with mean 1 and $(s, S) = (1, 3)$, the 4×4 transition probability matrix $\mathbf{P}$ is easily generated with a few lines of Maple code as I show in ▶`Periodic-sS.mw`◀. [In this case the state space is $\mathcal{S} = \{0, 1, 2, 3\}$ since shortages are not allowed.] Irreducibility of a DTMC is an important property that is rather difficult to ascertain by simple inspection.

I have a Maple code in ▶Irreducible-sS.mw◀ that implements an algorithm due to [7, p. 448] which checks this property. For the (s, S) inventory problem, we find that the chain is irreducible, i.e., has a single equivalence class (and ergodic). I then compute the stationary probabilities $(\pi_0, \ldots, \pi_3)$ by solving a system of four linear equations as shown in ▶Periodic-sS-Stationary.mw◀.

Another problem in Kao [12, p. 172–174] with 10 states is shown to have three equivalence classes $E_1 = \{1, 3\}$, $E_2 = \{2, 7, 9\}$ and $E_3 = \{6\}$, and four transient states $T = \{4, 5, 8, 10\}$ as presented in ▶Reducible-MC-Simpler-N10.mw◀. I also present an example from Isaacson and Madsen [10, p. 58] of a six-state DTMC which has two equivalence classes $E_1 = \{1, 3, 5\}$, and $E_2 = \{2, 6\}$, and one transient state $T = \{4\}$, see ▶Irreducible-Isaacson-Madsen-page-58.mw◀.

The n-step transition matrix $\mathbf{P}^{(n)}$ of a DTMC can be computed by finding $\mathbf{P}^n$. However, as n increases, the computation of $\mathbf{P}^n$ may become tedious. There exists an *exact* approach for computing the transient probabilities $\mathbf{P}^{(n)}$ that is based on generating functions which Maple can invert successfully. In ▶Transient-2by2.mw◀, I have a simple example of a 2-state DTMC whose transient probabilities are determined via the `rgf_expand()` function of Maple. This is achieved by inverting the elements of the generating function matrix $\mathbf{G}(z) = \mathbf{P}^{(0)}(\mathbf{I} - z\mathbf{P})^{-1}$, where $\mathbf{P}^{(0)}$ is defined as the identity matrix $\mathbf{I}$.

The mean first passage times (MFPT) μ_{ij}, $i, j \in \mathcal{S}$ are useful quantities in estimating the average length of time it takes for a DTMC to visit state j given that it started in state i at time 0. These mean times are obtained easily by solving a linear system of N^2 equations for a DTMC with N states. In ▶muij.mw◀, I return to the (s, S) inventory problem and compute the 16 MFPT μ_{ij}, $i, j \in \{0, 1, 2, 3\}$.

2.5 Continuous-time Markov Chains

Consider a continuous-time stochastic process $\{X(t), t \geq 0\}$ with the property that for $i, j \in \mathcal{S}$,

$$\begin{aligned} \Pr[X(t+s) = j \quad | \quad & X(s) = i,\, X(u) = x(u),\, 0 \leq u < s] \\ = \quad & \Pr[X(t+s) = j \mid X(s) = i] \end{aligned}$$

for all $s, t \geq 0$ and $x(u)$, $0 \leq u \leq s$. Such a process $\{X(t),\, t \geq 0\}$ is a continuous-time Markov chain (CTMC) with state space $\mathcal{S}$.

Similar to a discrete-time Markov chain, for a CTMC the conditional distribution of $X(t+s)$ given the past history over $0 \leq u \leq s$ depends only on the current state $X(s)$ at time s. The process $\{X(t),\, t \geq 0\}$ is a *time-homogeneous* CTMC if the conditional probability

$\Pr[X(t+s) = j \mid X(s) = i]$ is independent of s. In this case we write $\Pr[X(t+s) = j \mid X(s) = i] = p_{ij}(t)$ where $p_{ij}(t)$ is called the *transition function* from state i to state j. This quantity is analogous to the transition probability $p_{ij}^{(n)}$ of a discrete-time Markov chain. Defining the matrix of transition functions $\mathbf{P}(t) = [p_{ij}(t)]$, it can be shown that

$$\mathbf{P}'(t) = \mathbf{Q}\mathbf{P}(t), \quad \text{with } \mathbf{P}(0) = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{P}'(t) = [p'_{ij}(t)]$, and the diagonal elements of the ith row of the infinitesimal generator $\mathbf{Q}$ matrix is the negative of the sum of all other elements in row i. For example, for the Markovian queue with no waiting space (M/M/1/1), we obtain,

$$\begin{pmatrix} p'_{00}(t) & p'_{01}(t) \\ p'_{10}(t) & p'_{11}(t) \end{pmatrix} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix} \begin{pmatrix} p_{00}(t) & p_{01}(t) \\ p_{10}(t) & p_{11}(t) \end{pmatrix}. \tag{1}$$

First, note that if we had a *scalar* differential equation given by $p'(t) = qp(t)$, with $p(0) = 1$, the solution would simply be $p(t) = e^{qt} = \sum_{n=0}^{\infty}(qt)^n/n!$. It can be shown that for the matrix DE system $\mathbf{P}'(t) = \mathbf{Q}\mathbf{P}(t)$, $\mathbf{P}(0) = \mathbf{I}$, the solution assumes a similar form as $\mathbf{P}(t) = e^{\mathbf{Q}t}$, where $e^{\mathbf{Q}t} = \sum_{n=0}^{\infty}(\mathbf{Q}t)^n/n!$. Loading `with(LinearAlgebra)`, the function `MatrixExponential(Q, t)` conveniently solves the matrix DE and produces the transient solution for the transition functions $p_{ij}(t)$, $i, j \in \mathcal{S}$. The simplest example of the use of `MatrixExponential()` is the solution of the system of DEs given by (1) as shown in the Maple file ▶`MatrixExponential-MM11.mw`◀.

A more challenging problem (of a shoeshine shop) with three states and the exact solution of the transition functions is provided in the file ▶`MatrixExponential-Shoeshine.mw`◀.

The transient solution to the unconditional probabilities $p_j(t)$, $j \in \mathcal{S}$ in the M/M/1 queue involves the modified Bessel function of the 1st kind, and it is calculated in the Maple worksheet ▶`Transient-MM1.mw`◀.

In most problems it is sufficient to find the limiting probabilities $\pi_j = \lim_{t\to\infty} p_{ij}(t)$ of a CTMC. (These probabilities will exist provided that the transition matrix of the embedded Markov chain for the CTMC is irreducible and positive recurrent.) To evaluate the limiting probabilities, we establish the balance equations, i.e., for any state $j \in \mathcal{S}$, we write, **Output** rate from j = **Input** rate to j, which give rise to a system of N linear equations plus the equation $\sum_{j\in\mathcal{S}} \pi_j = 1$. (One of the structural equations can be eliminated as the original system is linearly dependent.) For example, for the M/M/1/1 queue described above, the

balance equations result in the following linear equations for each state $j = 0, 1$:

	Rate Out	=	Rate In
State 0:	$\lambda\pi_0$	=	$\mu\pi_1$
State 1:	$\mu\pi_1$	=	$\lambda\pi_0$.

Including $\pi_0 + \pi_1 = 1$ and solving, gives $[\pi_0, \pi_1] = \left[\frac{\mu}{\lambda+\mu}, \frac{\lambda}{\lambda+\mu}\right]$ for which Maple is not needed.

Returning to the three-state (shoeshine) example, the limiting probabilities are easily obtained in the Maple file ▶`Shoeshine.mw`◀. A generalized version of this problem with extra capacity and five states becomes more complicated but Maple solves for the limiting probabilities easily in ▶`Shoeshine-Extra.mw`◀.

A truly challenging problem concerned with the ambulance use in two response areas involves 9 states. The vector process $(X_1(t), X_2(t))$ is a CTMC with state space $\mathcal{S} = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2)\}$. This model is originally due to Carter, Chaiken and Ignall [4], but it is also discussed in Tijms [35]. After presenting this model, Tijms [35, p. 116] states (somewhat pessimistically): "These linear equations must be solved *numerically*." But Maple succeeds in solving the linear system *symbolically* (i.e., exactly!) in ▶`Ambulances.mw`◀. This problem is also discussed in Kao [12, Examples 5.1.3 and 5.2.3].

2.6 Brownian Motion

When time permits, I provide a brief discussion of Brownian motion (BM) based on my paper [27]. The Maple code in my paper was written in the early 2000s when Maple did not provide a function for simulating the Brownian motion process. This is now available as a function in the `Finance` package.

The function `BrownianMotion(x[0],mu,sigma,t)` simulates the stochastic differential equation $dX(t) = \mu(t)\ dt + \sigma(t)\ dW(t)$ for the BM $X(t)$ where $\mu(t)$ is drift and $\sigma(t)$ is the volatility and $W(t)$ is the standard Wiener process. The Maple file ▶`BM-Simulate.mw`◀ provides a simple example.

2.7 Innovative Use of Maple in the Stochastic Processes Course

How can Maple help in the effective teaching of a graduate course in stochastic processes? As I showed in several examples above, Maple can be useful in solving many problems analytically (i.e., in closed-form) which were not previously possible. For example, the ambulance problem discussed which was originally solvable only "*numerically*," can now be

solved by Maple *symbolically* (i.e., exactly!). Such results allow the researcher to conduct sensitivity analyses on the problem parameters without resorting to numerical techniques.

Maple's ability to invert a large class of Laplace transforms symbolically makes it possible to analyze challenging problems in renewal theory and continuous-time Markov chains. In certain cases where symbolic inversion is impossible, Maple can now perform the inversion numerically which provides an additional level of convenience in solving such problems.

Maple can also perform symbolic manipulation of random variables which is a non-trivial problem. Finding the probability density function (p.d.f.) of a sum of i.i.d. uniform random variables was illustrated above. Maple can also find the p.d.f. of a sum of exponential random variables $X_1, X_2, \ldots, X_n$ (for given n) with possibly different parameters $\lambda_1, \lambda_2, \ldots, \lambda_n$ which results in the generalized Erlang random variable. This problem can be solved in a manner similar to the steps used in the file ►`Three-Uniforms.mw`◄.

Maple's simulation capabilities are also impressive and I demonstrated them by showing that the superposition of a large number of renewal processes with large mean interarrival times is approximated by the Poisson process.

There are several problems which can be solved numerically with Maple using only a few lines of code, rather than a complicated set of commands using C++, Fortran, etc. Numerically inverting, or finding high powers of (large-scale) matrices, and simulating the Brownian motion process are such examples.

Thus, Maple's ability to perform symbolics, numerics and graphics makes it an ideal tool for teaching stochastic processes and provide innovative tools and techniques for dealing with a large class of problems arising in this course.

3 Dynamic Programming and Optimal Control (PhD course)

In this course I cover deterministic and stochastic dynamic optimization problems solved by dynamic programming and optimal control theory. Books by Bertsekas [3], Kamien and Schwartz [11] and Sethi and Thompson [33] can be consulted for rigorous expositions of these topics.

3.1 Deterministic Dynamic Programming

I start this course with a brief description of the simplest network model that lends itself to a dynamic programming (DP) formulation and solution. The "stagecoach" problem deals with a hypothetical 19th-century stagecoach company that transports passengers from California

to New York. Although the starting point (California) and the destination (New York) are fixed, the company can choose the intermediate **states** to visit in each **stage** of the trip. The solution with Maple is provided in ▶StageCoach.mw◀. For this problem, the value function V_{ij} is the minimum cost from any state i in stage j to the final state [New York] using the optimal policy. Given the state j in stage i, the decision to travel to state k in the next stage $i+1$ results in a cost (i.e., insurance premium) of $c_{ij}(k)$. Thus, the dynamic programming recursive equations are written as, $V_{51} = 0$ and $V_{ij} = \min_k \{c_{ij}(k) + V_{i+1,k}\}$, where the expression inside the parenthesis is minimized by a suitable choice of the decision variable k.

Models with linear system and quadratic cost are generally easily solved using the DP approach. For example, consider the sequential problem with the cost function $\sum_{t=0}^{2}(x_t^2 + u_t^2) + x_3^2$ and the system equations $x_{t+1} = x_t + u_t$, $t = 0, 1, 2$ with x_0 given as a constant. The Maple file with the solution of this problem is ▶LQ.mw◀. It is interesting to note that the optimal control law in such problems is always linear in the state variable x_t.

Forming the DP functional equation, we have

$$\begin{aligned} V_t(x_t) &= \min_{u_t}[x_t^2 + u_t^2 + V_{t+1}(x_t + u_t)], \quad t = 0, 1, 2 \\ V_3(x_3) &= x_3^2. \end{aligned}$$

We note that the sequential optimization model described above can also be solved as a standard nonlinear programming (NLP) problem with six decision variables $(\mathbf{x}, \mathbf{u}) = (x_1, x_2, x_3, u_0, u_1, u_2)$, the objective function $f(\mathbf{x}, \mathbf{u}) = \sum_{t=0}^{2}(x_t^2 + u_t^2) + x_3^2$ and three equality constraints $h_1 = x_1 - x_0 - u_0 = 0$, $h_2 = x_2 - x_1 - u_1 = 0$ and $h_3 = x_3 - x_2 - u_2 = 0$ with x_0 as a given constant; see ▶LQ-NLP.mw◀.

Naturally, in more general cases one could minimize $\sum_{t=0}^{N-1}(\mathbf{x}_t'\mathbf{Q}_t\mathbf{x}_t + \mathbf{u}_t'\mathbf{R}_t\mathbf{u}_t) + \mathbf{x}_N'\mathbf{Q}_N\mathbf{x}_N$ subject to $\mathbf{x}_{t+1} = \mathbf{A}_t\mathbf{x}_t + \mathbf{B}_t\mathbf{u}_t$, for $t = 0, 1, \ldots, N-1$, and $\mathbf{x}_0$ given, where $\mathbf{Q}_t$, $\mathbf{R}_t$, $\mathbf{A}_t$ and $\mathbf{B}_t$ are matrices and $\mathbf{x}_t$ and $\mathbf{u}_t$ are vectors (all appropriately dimensioned). Once again, the optimal control law is obtained as a linear function of the state vector $\mathbf{x}_t$ which requires the evaluation of the discrete Riccati equations. One such problem with a two-dimensional state vector $\mathbf{x}_t$ and a scalar control variable u_t is solved explicitly in terms of the discrete Riccati equations in ▶LinearRegulator.mw◀. Interestingly, Maple now has a `LinearAlgebra()` function named `DARE()` [Discrete Algebraic Riccati Equation] which solves the matrix Riccati equation. See ▶LinearRegulator-DARE.mw◀ for a comparison with the more cumbersome expressions in ▶LinearRegulator.mw◀.

3.2 Stochastic Dynamic Programming

One of the fruitful applications of stochastic dynamic programming is the optimality proof of the base stock policy for the periodic review inventory problem with stochastic demands. It can be shown that if the demands in each period are i.i.d., then it is optimal to order up to a level of S_n whenever the inventory level falls below S_n at the start of period n. In ▶BaseStock.mw◀ I illustrate this policy in a numerical example in a two-period problem.

In his seminal paper, Kelly [13] solves the following problem involving a gambler:

$$\begin{aligned} V_n(x_n) &= \max_{0\le u_n\le 1} E_{w_n} V_{n-1}(x_n + u_n x_n w_n) \\ &= \max_{0\le u_n\le 1} [pV_{n-1}(x_n + u_n x_n) + qV_{n-1}(x_n - u_n x_n)] \end{aligned}$$

with the boundary condition $V_0(x_0) = \log(x_0)$ where $V_n(x_n)$ as the maximum expected return if the gambler has a present fortune of x_n and has n gambles left. One of the most interesting features of this problem is the nature of its solution: The optimal strategy is myopic (invariant) in the sense that regardless of the number of bets left to place (n) and the current wealth (x), the optimal (nonnegative) fraction to bet in each period is the same, i.e., $u = p - q$ when $p > \frac{1}{2}$ and $u = 0$ when $p \le \frac{1}{2}$. I solve a specific example of this in ▶Gamble-Myopic.mw◀.

The final example involves the regulator problem discussed above with the added twist that the linear system is disturbed by the standardized normal i.i.d. r.v.'s w_t, i.e., $x_{t+1} = x_t + u_t + w_t$. The numerical example for this problem is given in ▶LinearRegulator-Scalar-Stochastic.mw◀.

3.3 Deterministic Optimal Control

In this part of the course, we consider optimal control problems with the objective functional $\int_0^T F[x(t), u(t), y]\ dt$ subject to the state equations $\dot{x}(t) = f[x(t), u(t), t]$ with the initial condition specified as $x(0) = x_0$ where $x(t)$ is the state variable and $u(t)$ is the control variable. In many problems, one encounters constraints on the control variable $g(u(t), t) \ge 0$, on the state variable $h(x(t), t) \ge 0$, or mixed-type constraints $k(x(t), u(t), t) \ge 0$ which complicate the problem. In some problems there could even be constraints on the final value of the state variable $x(T)$. These problems can be solved using either the Pontryagin's maximum principle or the Hamilton-Jacobi-Bellman equation; see, Sethi and Thompson [33, p. 347] for an excellent coverage of the techniques and business applications of optimal control theory.

One of the early examples in Sethi and Thompson [33, p. 42] is stated as max $J =$

$\int_0^2 (2x - 3u - u^2)\ dt$ s.t. $\dot{x} = x + u$, $x(0) = 5$ and $u(t) \in [0, 2]$. The optimal control is found as $u(t) = \text{sat}[0, 2; e^{2-t} - 2.5]$. The Maple file ▶Ex-2-5.mw◀ solves this "saturation-type" problem and also provides a numerical evaluation of the state trajectory $x(t)$ and the value of the objective functional.

One of the simple-looking but challenging problems in Sethi and Thompson [33, p. 48] has a quadratic objective but the system dynamics includes a cubic term for the state variable. The problem is to max $J = \int_0^1 -\frac{1}{2}(x^2 + u^2)\ dt$ s.t. $\dot{x} = x^3 + u$, with $x(0) = 5$. The two-point-boundary-value problem (TPBVP) arising from the necessary conditions are,

$$\begin{aligned} \dot{x} &= x^3 + u, \quad x(0) = 5, \\ \dot{\lambda} &= x + 3x^2\lambda, \quad \lambda(1) = 0. \end{aligned}$$

This is a nonlinear system of two ODEs and a closed-form solution seems not possible. Sethi and Thompson solve the TPBVP using Excel. I solved this problem with Maple in ▶Ex-2-7.mw◀ with the single command `dsolve(System union ICS, numeric, output = listprocedure, range = 0 .. 1)`. I also calculate the value of the objective as 0.8641.

In one of the earliest papers that uses control theory to solve a production problem, Hwang, Fan and Erickson [9] formulate their problem to

$$\max \int_0^T \left\{ -\frac{1}{2}c(u - \hat{u})^2 - \frac{1}{2}h(x - \hat{x})^2 \right\}\ dt$$

s.t. $\dot{x} = u - s$ where s is the constant sales rate. Applying Pontryagin's principle gives $u = \hat{u} + \lambda/c$ and $\dot{\lambda} = h(x - \hat{x})$ with $\lambda(T) = 0$ and $\dot{x} = \hat{u} + \lambda/c - s$ with $x(0) = x_0$. This is again a TPBVP but since we have a linear quadratic structure, it is possible to find the optimal solution for $u(t)$ and $x(t)$ exactly with Maple's help. This is shown in ▶Hwang-EtAl-Constant-s.mw◀ where I also present a numerical example.

As a final example, let's consider the problem of maximizing $J = -\int_0^1 x\ dt$, s.t. $\dot{x} = u$, with $x(0) = 1$ and $u \in [-1, 1]$. Sethi and Thompson [33, p. 37] solve this problem explicitly using the Pontryagin's principle and find that the solution is bang-bang due to the bounds on u, i.e., $u(t) = \text{bang}[-1, 1; \lambda(t)]$ where $\lambda(t)$ is the co-state variable. In my course I first solve the problem explicitly but also show the students how it can be cast as a linear programming problem since the integrand and the system ODE are both linear. With a suitable discretization, I find the same result as in Sethi and Thompson, see, ▶Ex-2-1-LP.mw◀. It is interesting to note that the LP formulation of this control problem results in 202 decision variables and 303 constraints.

3.4 Stochastic Optimal Control

This is a short chapter. I first discuss two of my earlier papers ([24] on stochastic control and [27] on stochastic differential equations).

To illustrate the use of stochastic optimal control, I consider a stylized production planning model (Sethi and Thompson [33, p. 347] where the objective is to min $E\int_0^T (u_t^2+X_t^2)\,dt+BX_T$ subject to $dX_t=(u_t-s_t)\,dt+\sigma dZ_t$ with the initial condition $X_0=x_0$ as given. Defining $V(x,t)$ as the value function, i.e., the minimum cost from time t in state $x_t=x$ to the final time T given that the optimal policy is followed, the Hamilton-Jacobi-Bellman (HJB) equation is,

$$0=\max_u\left\{-(u^2+x^2+V_t+V_x(u-s)+\frac{1}{2}\sigma^2V_{xx}\right\},\quad V(x_T,T)=Bx_T.$$

Differentiating and solving for u gives $u=\frac{1}{2}V_x$ indicating that the optimal control law can be obtained if V_x can be determined. To that end, one assumes a quadratic form for the value function as $V(x,t)=Q(t)x^2+R(t)x+M(t)$. Computing V_t, V_x and V_{xx}, and inserting these into the HJB equation results in a system of three nonlinear ODEs as,

$$\begin{aligned}\dot{Q} &= 1-Q^2,\quad Q(T)=0\\ \dot{R} &= 2sQ-RQ,\quad R(T)=B\\ \dot{M} &= sR-\frac{1}{4}R^2-\sigma^2Q,\quad M(T)=0,\end{aligned}$$

where s is assumed constant. The numerical solution of the ODE system for a particular set of parameter values and the resulting value function at $t=0$, i.e., $V(x,0)=-0.76x^2+5.46x-16.10$ is presented in the Maple file ▶Stochastic-Control-Numerical-System.mw◀ This file also evaluates the expression for the control law at $t=0$ as $u(x,0)=-0.76x+2.73$.

3.5 Innovative Use of Maple in the Dynamic Optimization Course

Discrete-time dynamic programming problems require the solution of recursive optimization problems involving the value function. Maple's ability to perform symbolic solution of optimization problems allows the user to find the optimal policy in closed-form as in the stochastic dynamic inventory problem. Here, the base stock policy is optimal, but requires the solution of the non-trivial problem of finding the optimal order-up-to levels S_n. In most cases, this problem is solved by approximating it as an infinite horizon problem. But, Maple makes it possible to evaluate the value function as a function of the state variable for each

time and find the optimal S_n.

Maple is also ideally suited for solving optimal control problems where a system's evolution over time is represented by differential equation(s). The necessary conditions are usually represented as two-point-boundary-value problems involving nonlinear differential equations which, in many cases, Maple can successfully solve in closed-form. When closed-form solution is not possible, such problems can still be solved with a single Maple command `dsolve` command as was presented in ▶Ex-2-7.mw◀.

Thus, Maple's symbolic and numerical solution facilities provide innovative techniques for dealing with both dynamic programming and optimal control problems.

4 Game Theory and Decision Analysis (PhD course)

In this course I use two textbooks by Gibbons [8] and Peters [28]. I also refer to some of my papers on game theory including Leng and Parlar [15], [16], [17], [18], Parlar [25] and Wu and Parlar [37]. In addition to using Maple to solve problems, I also use the Gambit Software available from `http://www.gambit-project.org/`. This software is capable of solving finite games even with multiple players.

4.1 What's Game Theory?

I start the course by discussing a few problems which can be formulated using game-theoretic concepts. Peters [28, p. 3] has a simple example of a zero-sum game that takes place in the South Pacific in 1943. The American admiral Kenney wants to bomb the Japanese admiral Imamura's troops which need to be transported to New Guinea across the Bismarck Sea. Each have two options: Northern route or Southern route. The zero-sum game is modelled as

$$\begin{bmatrix} 2,-2 & 2,-2 \\ 1,-1 & 3,-3 \end{bmatrix}$$

where the row strategies for Kenney are North or South, and the column strategies for Imamura are also North or South. In this matrix the numbers correspond to the number of days Kenney can bomb and thus he must choose a strategy to maximize his payoff. As we have a zero-sum game, for Imamura the opposite is true. By simple inspection, it can be shown that saddle point for this game is at the first row and first column, i.e., (N,N) resulting in a payoff of 2 days of bombing for Kenney. Since zero-sum games can also be solved using linear programming, I show the class the file ▶BismarckSea-Battle.mw◀ which finds the

same solution.

I also present a few examples of non-zero sum games and find their solution manually. In the next section, such problems are discussed in greater detail.

Peters [28, p. 15] presents a simple example of a bargaining problem where the players are to share a divisible good of unit quantity. If Player 1 receives a fraction α, his utility is $u_1(\alpha) = \alpha$ and if Player 2 receives a fraction β, her utility is $u_2(\beta) = \sqrt{\beta}$. The bargaining solution requires maximizing the objective $u_1(\alpha)u_2(\beta)$ subject to the constraint $\alpha + \beta \leq 1$ and $\alpha, \beta \geq 0$, (Nash [21]). I solve this problem in ▶Bargaining-alpha-beta.mw◀ and find $\alpha = \frac{2}{3}$ and $\beta = \frac{1}{3}$.

As a final example of introduction to game theory, let's consider the Cournot game with two firms with production quantities of q_i, $i = 1.2$, respectively. The total production is $Q = q_1 + q_2$, and the market price is $p = a - Q$ where a is the maximum market price as total production approaches zero. Each firm's production cost is $c_i q_i$, $i = 1.2$, respectively. The profit functions are $P_i(q_1, q_2) = q_i(a - Q) - c_i q_i$ for $i = 1, 2$. The Nash equilibrium ([22],[23]) in this case is obtained by finding the best responses for each firm by differentiating the profit functions with respect to a firm's order quantity and equating to zero. Solving the resulting system of linear equations gives $q_1 = \frac{1}{3}(a - 2c_1 + c_2)$, $q_2 = \frac{1}{3}(a + c_1 - 2c_2)$ with the firms' profits found as $P_1 = \frac{1}{9}(a - 2c_1 + c_2)^2$, and $P_2 = \frac{1}{9}(a + c_1 - 2c_2)^2$. These results are shown in the Maple file ▶Cournot.mw◀.

4.2 Static Games of Complete Information (Nash equilibrium)

Since every finite zero-sum game can be solved using linear programming as I showed in ▶BismarckSea-Battle.mw◀ above, I do not discuss these games in the remainder of the course. The equilibrium solution for non-zero sum games requires an understanding of Nash's approach and I illustrate this through several examples. For simple finite games with only a few rows and columns, manual inspections can help identify the equilibria. For larger games one can use a quadratic programming (QP) formulation proposed by Mangasarian and Stone [19]. If $(\mathbf{A}, \mathbf{B})$ are the payoff matrices for Player 1 and 2, and $(\mathbf{x}, \mathbf{y})$ are vectors of mixed strategies, respectively, Mangasarian and Stone have shown that the solution of the following QP will produce a Nash equilibrium, if one exists. (If there are multiple equilibria, starting with different initial solutions gives rise to the other equilibrium points.)

$$\begin{aligned} &\max_{a,b,\mathbf{x},\mathbf{y}} \ \mathbf{x}'(\mathbf{A}+\mathbf{B})\mathbf{y} - a - b \\ \text{s.t. } \mathbf{A}\mathbf{y} - a\mathbf{u} \ &\leq \ \mathbf{0}, \quad \mathbf{B}\mathbf{x} - b\mathbf{v} \leq \mathbf{0}, \quad \mathbf{u}'\mathbf{x} = 1, \quad \mathbf{v}'\mathbf{y} = 1, \quad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{y} \geq \mathbf{0}, \end{aligned}$$

where $\mathbf{u} = (1, 1, \ldots, 1)'$, $\mathbf{v} = (1, 1, \ldots, 1)'$, and a and b are the equilibrium payoffs to Players 1 and 2, respectively. As an example, consider the payoff matrices for each player,

$$\mathbf{A} = \begin{bmatrix} 6 & 0 & 0 \\ 10 & 5 & 0 \\ 8 & 8 & 4 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 6 & 10 & 8 \\ 0 & 5 & 8 \\ 0 & 0 & 4 \end{bmatrix},$$

where the rows and columns correspond to the prices levels chosen by each player, i.e., High, Medium and Low, and the numbers are the payoffs. I solve this problem using Maple's `QPSolve()` function and the (unique and pure) Nash equilibrium is found as $\mathbf{x} = \mathbf{y} = (0, 0, 1)$ with $a = b = 4$ in ▶`Mangasarian-Stone-Pricing.mw`◀.

In many business problems involving non-zero sum games, the players can choose from a continuum of values. In such cases, Nash equilibrium can be found employing standard calculus techniques. For example, in a two-person game, if $f(x, y)$ and $g(x, y)$ are the payoffs (utilities) to each player where x and y are the possible continuous strategy choices, the Nash equilibrium (if it exists) is computed as follows: Player 1 solves $\max_x \ f(x, y)$ for each value of y to obtain his best response $R_1(y)$, and Player 2 solves $\max_y \ g(x, y)$ for each value of x, to obtain her best response $R_2(x)$. These imply that to find the equilibrium, one needs to solve a system of two (possibly nonlinear) equations given by

$$I_1(x, y) = \frac{\partial f(x, y)}{\partial x} = 0, \quad I_2(x, y) = \frac{\partial g(x, y)}{\partial y} = 0.$$

I illustrate this in the Maple file ▶`Continuous-Example.mw`◀ where $f(x, y) = -2x^2 + 5xy$ and $g(x, y) = -3y^2 + 2xy + y$. The equilibrium is found as $(x, y) = (\frac{5}{14}, \frac{4}{14})$ with payoffs $(f, g) = (0.255, 0.244)$. The Maple file also illustrated the intersection of the two best response relations.

Consider a simplified version of the competitive newsvendor model discussed in Parlar [25] and Wu and Parlar [37]. The newsvendors face random demands X and Y with respective densities $f(x)$ and $h(y)$ and if one newsvendor runs out of stock, some of the unsatisfied customers may switch to the other newsvendor if he/she has any units available. With these assumptions, the expected profit function of the first newsvendor (P1) in Wu and Parlar [37]

is given as,

$$\begin{aligned} J_1(q_1,q_2) &= s_1\int_0^{q_1} x f(x)\,dx + s_1 q_1 \int_{q_1}^{\infty} f(x)\,dx + s_1\int_0^{q_1}\int_{q_2}^{B} b(y-q_2)h(y)f(x)\,dy\,dx \\ &\quad + s_1\int_0^{q_1}\int_{B}^{\infty}(q_1-x)h(y)f(x)\,dy\,dx - c_1 q_1, \end{aligned}$$

where s_1 is the unit sales revenue, c_1 is the unit purchase cost and $B \equiv (q_1 - x)/b + q_2$, with b as the fraction of P2's demand that will switch to P1's product when P2 is sold out. The second newsvendor's expected profit is obtained similarly. After computing $\partial J_1/\partial q_1 \equiv I_1(q_1,q_2) = 0$ and $\partial J_2/\partial q_2 \equiv I_2(q_1,q_2) = 0$, I prove the uniqueness of the Nash equilibrium for this problem. The Maple file ▶Newsvendors-Nash-Stackelberg.mw◀ presents a numerical example with exponential demand densities. This file also includes a related example of the Stackelberg equilibrium where one newsvendor is the leader and the other a follower.

4.3 Dynamic Games of Complete Information (Subgame perfect equilibrium)

In the static games of complete information discussed in the previous section (chapter in the course) all players choose their strategies simultaneously and Nash is the solution concept for such problems. However, in many business problems decisions are often made sequentially and there may be a first mover (leader) and a second mover (follower). For example, in a two-person game, if $f(x,y)$ and $g(x,y)$ are the payoffs (utilities) to each player where x and y are the possible continuous strategy choices and leader optimizes $f(x,y)$ and the follower optimizes $g(x,y)$. Such problems are solved using the "backward induction" approach, sometimes also known as the Stackelberg strategy and give rise to the subgame perfect equilibrium. If the follower observes leader's strategy choice of x, then for any value of x, the follower solves $\max_y\ g(x,y)$ by finding her best response from $\partial g/\partial y = 0$ which results in a relation $I_2(x,y) = 0$ in the (x,y)-plane. Since the game is one of complete information, the leader can compute $I_2(x,y) = 0$ and optimize his objective subject to this relation. That is, the leader's problem becomes $\max_x\ f(x,y)$ s.t. $I_2(x,y) = 0$. In some cases where $I_2(x,y) = 0$ can be solved uniquely for $y = \phi(x)$, the leader's problem simplifies to $\max_x\ f(x,\phi(x))$.

I have a very simple example in the Maple file ▶Extensive-Continuous.mw◀ which solves the socially optimal, individual optimization, Nash and Stackelberg problems. Here, the objective for the leader's is to minimize $f(x,y) = (x-1)^2 + (y-1)^2$ and the objective of the follower is also to minimize $g(x,y) = 2(x-2)^2 + (y-2)^2 - 2xy + 40$.

The second and last example in this section again consider the newsvendor problem where

the Maple file ▶Newsvendors-Nash-Stackelberg.mw◀ considered above also computes the Stackelberg strategy and find that in this problem there is an advantage to the first mover (leader).

4.4 Static Games of Incomplete Information (Bayesian Nash equilibrium)

In the last two sections (and chapters in the course) it was assumed that the information about the players' payoff functions and other parameters was public knowledge. For example, in the Cournot game discussed in Section 4.1, both firms were privy to the value of the maximum market demand a, and their marginal production costs c_1 and c_2. In many business problems this may be an oversimplification of reality as one firm may not know the exact value of the other firm's marginal cost.

As a simple example of a situation where there is incomplete information, let's assume that Firm 1's production cost is $C_1(q_1) = cq_1$ with marginal cost of c, and both firms know this, (Gibbons [8, Ch. 3]). However, Firm 1 believes that Firm 2's marginal production cost is either c_H [with conditional probability $\Pr(c_H \mid c) = \theta$], or c_L [with conditional probability $\Pr(c_L \mid c) = (1-\theta)$] where $c_H > c_L$. Here, Firm 2 has an advantage over the other as it knows both Firm 1's marginal cost and its own, which is either c_H or c_L. The Bayesian Nash equilibrium is obtained by formulating and solving new objective functions for both firms as,

$$\begin{aligned}
\hat{P}_1(q_1; q_{2H}, q_{2L}) &= \theta\{[a-(q_1+q_{2H})]-c\}q_1 + (1-\theta)\{[a-(q_1+q_{2L})]-c\}q_1, \\
\hat{P}_{2H}(q_1; q_{2H}) &= \{[a-(q_1+q_{2H})]-c_H\}q_{2H}, \\
\hat{P}_{2L}(q_1; q_{2L}) &= \{[a-(q_1+q_{2L})]-c_L\}q_{2L},
\end{aligned}$$

where q_{2H} and q_{2L} are the production quantities for Firm 2 when its marginal cost is High, or Low, respectively. Solving

$$\frac{\partial \hat{P}_1}{\partial q_1} = 0, \quad \frac{\partial \hat{P}_2}{\partial q_{2H}} = 0, \quad \frac{\partial \hat{P}_2}{\partial q_{2L}} = 0,$$

gives the Bayesian Nash equilibrium quantities for both firms as,

$$\begin{aligned} q_1 &= \frac{1}{3}\{a - 2c + [\theta c_H + (1-\theta)c_L]\}, \\ q_{2H} &= \frac{1}{3}(a - 2c_H + c) + \frac{1-\theta}{6}(c_H - c_L), \\ q_{2L} &= \frac{1}{3}(a - 2c_L + c) - \frac{\theta}{6}(c_H - c_L). \end{aligned}$$

These results are shown in the Maple file ▶Cournot-Incomplete.mw◀.

The competitive newsvendor problem discussed in Section 4.2 can also be formulated assuming incomplete information. The Bayesian Nash equilibrium for this version of the problem is provided in the Maple file ▶Newsvendor-Incomplete.mw◀, also covered in Wu and Parlar [37].

4.5 Mechanism Design (Adverse Selection)

In the last three sections we used the Nash, subgame perfect and Bayesian Nash solution concepts to find the equilibria of the resulting game models. These solution concepts have found wide applicability in many business applications, but they lack an incentive mechanism for a player (principal) to influence the actions of another (agent). In mechanism design formulations of multi-player interactions, this is made possible.

In this section I will describe the adverse selection formulation of the interaction between a principal who uses a menu of choices for the agent so that the latter's incentives are aligned with the former. In such problems, the agent has hidden knowledge/information about certain aspects of a problem, such as her marginal cost, and the principal will design an incentive system so that the agent will reveal her information. For excellent coverages of these ideas, see, Dixit, Skeath and Reiley [5, Ch. 14] and Laffont and Martimort [14].

Consider a simple example where the agent could either be a low-cost, or high-cost type with the marginal cost θ_1 or θ_2, respectively. The principal knows that the agent is low-cost with probability ν and high cost with probability $1-\nu$. The agent produces a good for the principal for whom the utility (or, social value function) is $S(q)$ for q units of production. Since the principal does not know the agent's type, he will offer her a menu $[(t_1, q_1), (t_2, q_2)]$ and pay her t_1 for producing q_1 units and t_2 for q_2 units. The principal's problem is to choose the optimal values of the menu parameters to maximize his expected utility of $\nu[S(q_1) - t_1] + (1-\nu)[S(q_2) - t_2]$. However, the principal must assure that the agent participate in this arrangement so two participation constraints (PC) must be satisfied as $t_1 - \theta_1 q_1 \geq 0$, and $t_2 - \theta_2 q_2 \geq 0$. Even thought these PCs are necessary, they are not sufficient for a complete formulation of the adverse selection problem. The principal also

must ensure that the agent does not "lie" about her marginal cost.

If the agent is low-cost, then her profit will be $t_1 - \theta_1 q_1$. On the other hand, if she claims to be high-cost when she is in fact low cost, then her profit is $t_2 - \theta_1 q_2$. The principal should therefore add a new incentive compatibility (IC) constraint $t_2 - \theta_1 q_2 \geq t_2 - \theta_1 q_2$ so that the agent is incentivized to report her true (low-) cost. Similarly, for completeness, the constraint $t_2 - \theta_2 q_2 \geq t_1 - \theta_2 q_1$ must also be included so that a high-cost agent does not pretend to be low-cost (which would not be beneficial, in any case). Thus, the principal's problem is to maximize the expected utility subject to the two PC and two IC constraints.

If we now define information rents paid to the agent as $U_1 = t_1 - \theta_1 q_1$, $U_2 = t_2 - \theta_2 q_2$, when she is low-cost and high-cost, respectively, the problem can be re-stated as,

$$\max_{U_i, q_i, i=1,2} \underbrace{\nu[S(q_1) - \theta_1 q_1] + (1-\nu)[S(q_2) - \theta_2 q_2]}_{\text{Expected utility}} - \underbrace{[\nu U_1 + (1-\nu) U_2]}_{\text{Information rent}}$$

$$\begin{aligned} \text{s.t. } U_1 &\geq U_2 + q_2 \Delta\theta \quad \text{(IC1)} \\ U_2 &\geq U_1 - q_1 \Delta\theta \quad \text{(IC2)} \\ U_1 &\geq 0 \quad \text{(PC1)} \\ U_2 &\geq 0 \quad \text{(PC2)} \end{aligned}$$

where $\Delta\theta = \theta_2 - \theta_1$. Employing a few simple economic arguments, it can be shown that $U_1 = q_2 \Delta\theta$ and $U_2 = 0$. That is, the low-cost agent receives a positive information rent, whereas the high-cost one does not. For a numerical example of the complete model, see ►SecondBest-AdverseSelection.mw◄. In this example we set $S(q) = 15q - \frac{1}{2}q^2$, $(\theta_1, \theta_2) = (3, 5)$, and $\nu = \frac{2}{3}$, and find $(U_1, U_2) = (12, 0)$, with $(q_1, q_2) = (12, 6)$. This result shows that the low-cost agent would have a positive information rent of $U_1 = 12$ and receive a larger order than a high-cost agent would.

4.6 Cooperative Games with Transferable Utility

In all the game-theoretic models presented so far (zero-sum and nonzero-sum), the players had no reason, or incentive, to cooperate. However, in many realistic problems, especially those involving supply chains (Leng and Parlar [17]), the players can cooperate and this can result in improved performance for the system, i.e., "grand coalition." The relevant question now is to find a reasonable method of splitting the added benefits (savings, profits, etc.) so that the whole group of players will continue cooperating.

In this section, I will present three concepts that are crucial to the analysis of cooperative

games. I will first explain the concept of core, which may be empty or non-empty. I will then discuss Shapley value as one possible solution concept for splitting the additional benefits resulting from cooperation. Finally, I will present another important solution concept known as the nucleolus. I will base my discussion on an example in Peters [28, p. 12].

4.6.1 Core

A power source serves three cities denoted by $N = \{1, 2, 3\}$ which are at different distances from the source. The cities can rent transmission links and if they cooperate, they can save on costs of reaching the power source. Cities can establish coalitions for cooperation, defined as subsets S of the set N. In the example, it is shown that the cost savings for different coalitions are $v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$, $v(\{1, 2\}) = 90$, $v(\{1, 3\}) = 100$, $v(\{2, 3\}) = 120$ and for the grand coalition $v(\{1, 2, 3\}) = 220$, where $v(\cdot)$ is the characteristic function of the game. To determine how much each player (city) should gain from these savings, we define x_i as the payoff (imputation) to player $i \in S$. What should be a "reasonable" set of allocations to each player? Since $\{1, 2\}$ can already save 90 if they cooperate, to keep them in the game, the total sum of payments to these two players should be at least 90, i.e., $x_1 + x_2 \geq 90$. Similarly, we have $x_1+x_3 \geq 100$, $x_2+x_3 \geq 120$ and for the grand coalition $x_1+x_2+x_3 = 220$. The core, is thus defined as the set $\mathcal{C} = \{(x_1, x_2, x_3) : x_i \geq 0,\ i = 1, 2, 3$ and $x_1 + x_3 \geq 100$, $x_2 + x_3 \geq 120$, $x_1 + x_2 + x_3 = 220\}$. In the Maple file ▶`CoreGraph-ThreeCities.mw`◀, the core plotted in three dimensions as the middle section of the red plane bounded by three planes arising from the boundary of the inequalities.

In some problems the core may be empty. It is easy to check this property by solving a simple linear programming problem with the constraints arising from the coalitions' imputation constraints and minimizing the objective as, say, x_1. If the feasible set to the `LPSolve()` function exists, then the core is non-empty. Otherwise, if `LPSolve()` can't find a feasible solution, then the core must be empty. In our problem, the core is non-empty and this is shown in the Maple file ▶`CoreTest-ThreeCities.mw`◀. This file has two other example with empty cores; one is the well-known Divide the Dollar game where $v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$, $v(\{1, 2\}) = v(\{1, 3\}) =$, $v(\{2, 3\}) = 1$ and for the grand coalition $v(\{1, 2, 3\}) = 1$.

4.6.2 Shapley value

This is one of the earliest solution concepts in cooperative game theory due to Shapley [34]. The essential idea is to assign to each player the "average marginal contribution" she makes to a coalition. For small number of players a simple manual procedure can be used to compute the Shapley value. However, in general, Shapley value for player i in the cooperative

game (N, v) can be computed from,

$$\Phi_i(N, v) = \sum_{S \subseteq N,\ i \notin S} \frac{|S|!(n - |S| - 1)![v(S \cup \{i\}) - v(S)]}{n!},$$

where n is the number of players in grand coalition N, and $|S|$ is the number of players in coalition S. For the three cities example presented above, to find the Shapley value for Player 1, the term inside the summation for $S = \{2\}$ is $90/6 = 15$. Similarly, for $S = \{3\}$, the term inside the summation is $100/6$ and for $S = \{2, 3\}$ it is $100/3$. Summing these terms, we obtain $\Phi_1(N, v) = 65$ as the Shapley value for Player 1. It is important to note that even when the core is non-empty, Shapley value may be outside the core. The Maple file ▶Shapley-Baron-ThreeCities.mw◀ (due to Barron [2, Ch. 5]) computes the Shapley value for any cooperative game with three players, in particular the example with three cities discussed above for which we find $\mathbf{\Phi} = (65, 75, 80)$. The number of players can be easily increased by adjusting the value of the parameter N, and by entering the relevant values of the characteristic functions for any finite number of players.

4.6.3 Nucleolus

Recall that imputations in the core should satisfy the constraints $\sum_{i \in S} x_i \geq v(S)$ for any coalition $S \subseteq N$. But in some cases the core may be empty (as in the Divide the Dollar game mentioned above). If so, one can try to satisfy the constraints as much as possible by making the largest violation as small as possible; see, Schmeidler [32]. To that end, we define the "excess" (unhappiness) of S at the imputation vector $\mathbf{x}$ as,

$$e_S(\mathbf{x}) = v(S) - \sum_{i \in S} x_i,$$

and make the most unhappy coalition as little unhappy as possible. Unlike the Shapley value for which a closed-form formula for the imputations exists, computation of the nucleolus requires an iterative approach for small problems, and the solution of a sequence of linear programming problems for larger problems. However, the paper by Leng and Parlar [18] provides closed-form formulas for nucleolus in any three-person cooperative game. These formulas are incorporated into the Maple file ▶Nucleolus-ThreeCities.mw◀ which computes the nucleolus for the three cities problem and find the imputation as $x = (56\frac{2}{3}, 76\frac{2}{3}, 86\frac{2}{3})$ which is slightly different from the Shapley value found in the same example.

4.7 Innovative Use of Maple in the Game Theory Course

Maple's symbolics, numerics and graphics capabilities have made it an ideal computer algebra system for teaching introductory mathematics courses and it has been adopted at numerous science and engineering departments. To my knowledge, Maple is not widely-known by academics who teach in economics and social sciences departments. However, game theory is a popular course taught at undergraduate- and graduate-levels in almost every economics department and the tools I presented in this section may be of benefit to instructors teaching game theory courses.

In particular, the explicit results found in the Bayesian Nash equilibrium model, simplicity of computing the Shapley value and the simple modelling of the mechanism design problems as shown in this section would make Maple a mathematical tool of innovation and value to the economists teaching game theory.

It is also worth mentioning that two of the chapters in Parlar [26] covered stochastic process and dynamic programming. However, game theory is not included in this book which would make the contents of this section especially relevant in a game theory course.

5 Conclusions

Maple is one of the most popular and sophisticated computer algebra systems with a knowledge base that includes, among others, geometry, calculus, differential equations, transform techniques and optimization.

These features of Maple have made it an invaluable tool in the PhD-level management science courses I have taught in the business school of my home university. Maple's ability to perform symbolic manipulations, such as differentiation and integration, and solving (systems of) differential equations makes it possible to go beyond the "toy" problems and present realistic examples of management science problems in class. Maple is also capable of performing sophisticated numerical computations and generating publication-quality graphics, such as 3-D surfaces and contour plots.

The topic of the present special issue is "Model Development for the Classroom" and its purpose is to "present a collection of articles discussing the innovative use of model development environments [e.g., Maple] in courses held at colleges and universities." In this paper I summarize my experience of using Maple in three of my PhD courses, i.e., stochastic processes, dynamic programming and optimal control, and game theory. It is my hope that the examples and cases I have presented here will be of benefit to the instructors of PhD-level (or, other advanced) management science/operations research courses taught in business

schools, industrial engineering, and mathematics/statistics departments. I am confident that Maple can also be successfully used in courses on queueing theory, nonlinear programming, scheduling, probability and mathematical statistics, linear algebra and operations modelling, providing realism in the examples and topics covered in these courses.

Conflict of interest statement: The corresponding author states that there is no conflict of interest.

Data Availability Statement: Data sharing not applicable to this article as no datasets were generated or analysed during the current study. However, the Maple files cited will be available on the author's homepage at `https://profs.degroote.mcmaster.ca/ads/parlar/ORMapleBook/Parlar-Supplements-SpringerORForum.zip`.

References

[1] J. Abate and W. Whitt. A unified framework for numerically inverting Laplace transforms. *INFORMS Journal on Computing*, 18(4):408–421, 2006.

[2] E. N. Barron. *Game Theory: An Introduction*. Wiley-Interscience, 2007.

[3] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Englewood Cliffs, 1987.

[4] G. Carter, J. M. Chaiken, and E. J. Ignall. Response areas for two emergency units. *Operations Research*, 20:571–594, 1972.

[5] A. Dixit, S. Skeath, and D. Reiley. *Games of Strategy*. W. W. Norton & Company, New York, 3rd edition, 2009.

[6] S. Edwards. A computer-algebra-based calculating system. *BYTE*, 8(12):481, 1983.

[7] D. P. Gaver and G. L. Thompson. *Programming and Probability Models in Operations Research*. Brooks/Cole, Monterey, Calif., 1973.

[8] R. Gibbons. *Game Theory for Applied Economists*. Princeton University Press, Princeton, New Jersey, 1992.

[9] C. L. Hwang, L. T. Fan, and L. E. Erickson. Optimum production planning by the maximum principle. *Management Science*, 13(9):751–755, 1967.

[10] D. L. Isaacson and R. W. Madsen. *Markov chains: Theory and applications*. Wiley, 1976.

[11] M. I. Kamien and N. L. Schwartz. *Dynamic optimization: the calculus of variations and optimal control in economics and management*. Elsevier, Amsterdam, 1991.

[12] E. P. C. Kao. *An Introduction to Stochastic Processes*. Duxbury, Belmont, California, 1997.

[13] J. L. Kelly. A new interpretation of information rate. *Bell System Technical Journal*, 35:917–926, 1956.

[14] J.-J. Laffont and D. Martimort. *The Theory of Incentives: The Principal-Agent Model*. Princeton University Press, Princeton, New Jersey, 2002.

[15] M. Leng and M. Parlar. Game theoretic applications in supply chain management: a review. *INFOR*, 43(3):187–220, August 2005.

[16] M. Leng and M. Parlar. Game-theoretic analysis of an ancient Chinese horse race problem. *Computers and Operations Research*, 33:2033–2055, 2006.

[17] M. Leng and M. Parlar. Allocation of cost savings in a three-level supply chain with demand information sharing: A cooperative-game approach. *Operations Research*, 57(1):200–213, January–February 2009.

[18] M. Leng and M. Parlar. Analytic solution for the nucleolus of a three-player cooperative game. *Naval Research Logistics*, 57:667–672, 2010.

[19] O. L. Mangasarian and H. Stone. Two-person nonzero-sum games and quadratic programming. *Journal of Mathematical Analysis and Applications*, 9(3):348–355, 1964.

[20] J. Medhi. *Stochastic Processes*. John Wiley, New York, 1994.

[21] J. Nash. The bargaining problem. *Econometrica*, 18:155–162, 1950.

[22] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36:48–49, 1950.

[23] J. Nash. Non-cooperative games. *The Annals of Mathematics, Second Series*, 54(2):286–295, September 1951.

[24] M. Parlar. Use of stochastic control theory to model a forest management system. *Applied mathematical modelling*, 9(2):125–130, 1985.

[25] M. Parlar. Game theoretic analysis of the substitutable product inventory problem with random demands. *Naval Research Logistics*, 35(3):397–409, 1988.

[26] M. Parlar. *Interactive Operations Research with Maple: Methods and Models*. Birkhäuser, Boston, 2000.

[27] M. Parlar. A simplified treatment of Brownian motion and stochastic differential equations arising in financial mathematics. *PRIMUS: Problems, Resources, and Issues in Mathematics Undergraduate Studies*, 14(3):269–287, 2004.

[28] H. Peters. *Game Theory: A Multi-Leveled Approach*. Springer-Verlag, Berlin, 2008.

[29] J. D. Pintér. *Computational global optimization in nonlinear systems: an interactive tutorial*. Lionheart publishing, 2001.

[30] J. D. Pintér. *Global optimization: scientific and engineering case studies*, volume 85. Springer Science & Business Media, 2006.

[31] S. Ross. *Stochastic Processes*. John Wiley, New York, 1983.

[32] D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, 17:1163–1170, 1969.

[33] S. P. Sethi and G. L. Thompson. *Optimal Control Theory: Applications to Management Science and Economics*. Springer, New York, 2nd edition, 2000.

[34] L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[35] H. C. Tijms. *Stochastic Modeling and Analysis*. John Wiley, Chichester, 1986.

[36] G. Williams. The muSIMP/muMATH-79 symbolic math system, a review. *BYTE*, page 324, Nov. 1980.

[37] H. Wu and M. Parlar. Games with incomplete information: A simplified exposition with inventory management applications. *International Journal of Production Economics*, 133:562–577, 2011.