APL 82, W. H. Janko, W. Stucky (ed.),
APL Quote Quad, Vol. 13 No. 1,

A CONCEPTUAL FRAMEWORK FOR THE DESIGN OF MARKET SIMULATIONS

Girard C. Pessis
Associate Professor
Faculty of Business Administration
The University of Alberta
Edmonton, Alberta, Canada T6G 2G1

Abstract

A conceptual framework for the design of Market Simulations is developed using a systematic modularization concept in APL to allow easy and flexible combination of data entry and computational function. The Framework involves the definition of a Data Bank, an Executive Processor, and a method for defining functional relationships. It combines the power of APL with the ease of application characteristic of specialized simulation languages. Two important features of the conceptual framework are strict adherence to a predefined array syntax for functions representing market components, and a method of modifying simulations involving the use of executable lists. A protocol is developed which employs the syntactical structure of APL, and a demonstration is provided.

I. Introduction

The purpose of this research is to define a set of programming conventions using a systematic modularization concept in APL to improve the design and implementation of marketing simulations, and provide for more effective problem specification and sensitivity analysis. Models using the conceptual framework have been developed and programmed for courses and labs dealing with Decision Simulation in Marketing. One such model, an extension and modification of a Competitive Strategy Simulation,' will be discussed to demonstrate the power of this approach to simulation. All functions for the model are written in APL. It must be emphasized that the conceptual framework proposed here need not be limited to marketing simulations or any other particular type of simulation model. As a methodology for model specification and programming, this framework has a wide range of possible applications.

II. Systematic Modularization Approach to Simulation

A simulation can be modularized into the following components:

1. A Data Bank,
2. Functions which enter data into the Data Bank,
3. Functions which retrieve data, and
4. Functions which analyse the retrieved data.

The Proposed framework requires that a simulation model must be broken down into fundamental parts; i.e., modularized into as many distinct pieces as possible, and grouped according to the four components given above. Each component must be assigned a structure. Functions within each component must have a consistent syntax, and protocols must be developed to link all parts. The virtue of the systematic modularization approach to simulation is that it can be used and modified easily. The conceptual framework developed in this paper employs this approach and the syntactical structure of APL to achieve substantial design time and programming economies.

There are two basic types of languages for programming simulations: a general purpose language such as APL, Fortran, PL1, and BASIC; and a special purpose simulation language such as SIMSCRIPT and GPSS. With general purpose languages one has complete flexibility but everything must be designed and specified by the programmer. Simulation languages such as GPSS allow for quick programming but are only useful for specific classes of simulation problems; e.g., scheduling, waiting line, and inventory. A programming framework in which all concepts are modularized allows one to combine the flexibility and power of a general purpose programming language (APL) with the ease of application of languages such as GPSS and SIMSCRIPT.

III. Framework Components

A. Data Bank

The Data Bank shall consist of k n-dimensional arrays, where each array represents a different marketing instrument, and each dimension represents a descriptor for each marketing instrument. For example, A[I;J;T;M] represents advertising expenditures for firm I, product J, time T, and market M. Data is organized into n-dimensional arrays to take advantage of the array structure and syntax of APL. All simulated results are produced by functions which use as inputs the arrays in the Data Bank.

The syntax for all analysis functions, data manipulation functions, and the user interface is designed to take advantage of the array structure of the Data Bank. This is a key element of the conceptual framework. In the marketing simulation example to follow, the data bank consists of 3 arrays called P, A, and D for the price, advertising, and distribution marketing instruments over time and firm. All functions producing simulated results must conform to a predetermined syntax, and use the arrays in the data bank as inputs.

B. Structural Functions

A Market Simulation requires functions which represent the component parts of the market. These functions not only represent the behavior of firms and consumers in the market, but also the characteristics of the market such as market demand, channel logistics, growth, labor force, communications, etc. A Structural Function is defined as a function which models the behavior of a single market component. For example, each firm within the market can be represented by many different functions such as cost functions, profit functions, demand functions, inventory functions, etc. It is imperative that each function be distinctly defined; i.e., a separate subroutine programmed for each Structural Function. This practice of modularization; i.e., the use of subroutines, has long been a common practice among programmers, and with complex simulations this practice can lead to confusion. It is therefore necessary that syntactical conventions be specified, and functions of similar types be grouped into syntactically consistent sets. All profit functions, all cost functions, etc. must have the same syntactical structure, and use as inputs elements from the Data Bank. The Structural Functions, defined in such a manner, thus become the building blocks for the simulation, and the actual programming of the simulation is reduced to a process of specifying combinations of these Structural Functions. This provides tremendous flexibility for model specification and sensitivity analysis. Since all Structural Functions depend upon the arrays in the Data Bank, one must specify values for these arrays. This is accomplished by Discretionary Functions.

C. Discretionary Functions

While the Structural Functions represent the market structure and characteristics of companys within the market, the Discretionary Functions are defined as functions which set values of decision variables. These functions represent specific actions or strategies of participants in the market. They may consist of pricing strategies, channel and logistic strategies, promotional strategies, and product development strategies. Market Simulations compare the evolution of a market under a set of benchmark strategies with the evolution of the market under alternative strategies. The conventions proposed allow for not only modification of existing Discretionary Functions, but the introduction of new functions.

Each Discretionary Function is represented by a unique function name which appears in a list given to the executive routine as an argument (input). Since all Discretionary Functions must have a common syntax, the function name list can be converted into a list of function calls and executed in a single sequence. If a function name does not appear in the list, the function represented is not executed. Since the executive routine treats every element in the list in the same manner, additional functions can be added or deleted without modifying other parts of the simulation. The programmer need only be concerned with programming one strategy at a time, and need not be concerned with the interface between that strategy and the rest of the simulation. The commonality of syntax, the storage of values of decision variables in the Data Bank, and the execution of functions according to a function name list, provides all the necessary linkages between components of the simulation.

D. Executive Processor

The Executive Processor is the master program which manages the sequence of steps necessary to run the simulation. If the proposed framework is followed, the Executive Processor will resemble an outline of the total simulation. It will contain documentation and all the steps necessary to execute the Structural and Discretionary Functions. It must be provided with a function name list for Discretionary Functions either as a global variable or as an argument. The Executive Processor creates the Data Bank, calls Structural Functions, calls Discretionary Functions, and provides output.

The Executive Processor can also be programmed with a second argument list representing the set of Structural Functions. This is desirable if the market is to be studied under alternative structures. This second list must contain the names of variables containing the canonical representation of each Structural Function. Different sets of Structural Functions can be tested by specifying each list in turn. Simulating different market characteristics thus involves the following:

1. Creating lists representing names of programmed Structural and Discretionary Functions,
2. Having all necessary Structural and Discretionary Functions present in the active workspace, and
3. Creating and executing an Executive Processor whose arguments are matrices containing lists of Structural and Discretionary Function names.

IV. Competitive Strategy Model

The Model used to demonstrate the conceptual framework involves a market with 2 competing firms. It deals with "the problem of formulating a long-run competitive marketing strategy for a new product introduced into a market with classic growth, seasonal and merchandising characteristics."[2] The original profit accounting model had 8 Structural Functions, and the original Competitive Strategy Model involved 13 different Discretionary Strategy Functions that each competitor can choose from.[3] The benchmark conditions assume that each competitor has the same history and operates under the same Structural Functions. Each strategy is compared pairwise with every other strategy over a fixed time horizon in order to analyze a duopoly confrontation.

This section will present APL expressions for each Structural Function, each Discretionary Function, and an Executive Processor. Since the model chosen for exposition is a marketing strategy model involving few functions, a list structure involving only a single list for Discretionary Functions is used, and all functions are coded in function format rather than in canonical form.

A. Structural Functions

Each Structural Function is defined with the following properties:

1. A Common Syntax *R←SFNAME T*
2. A dependence upon elements of the Data Bank (Prices, Advertising Expenditures, and Distributional Expenditures) which are generated using the Discretionary Functions,
3. The global variable *N* specifying the number of time periods, and
4. Other Structural Functions.

Each Structural Function has a name *SFNAME*, uses the global variable *T* (the time periods on which the calculation is based) as an argument, and returns the result *R* which in the case of 2 competitors is a matrix with 2 rows and as many columns as elements in *T* . For example, to find Sales Levels for the month of January for 3 successive years, one would use the Sales function *SALES* 1 13 25, where January is the first month in the simulation and monthly data is assumed.

The eight Structural Functions are as follows:

1. Profit Function

```
    ∇ PR←PROFIT T
[1]  ⍝ PR IS A MATRIX OF PROFITS WHERE
[2]  ⍝ PR[I;J] IS THE PROFIT OF THE
[3]  ⍝ ITH FIRM IN PERIOD T[J].
[4]   PR←(P[;T]×SALES T)-COST T←,T
    ∇
```

2. Cost Function

```
    ∇ C←COST T
[1]  ⍝ C IS A MATRIX OF COSTS WHERE C[I;J]
[2]  ⍝ IS THE COST OF THE ITH FIRM
[3]  ⍝ IN PERIOD T[J].
[4]   C←3167+A[;T]+D[;T]+10×SALES T←,T
    ∇
```

3. Sales Function

```
    ∇ S←SALES T;M
[1]  ⍝ S IS A MATRIX OF SALES WHERE S[I;J]
[2]  ⍝ IS THE SALES OF THE ITH FIRM
[3]  ⍝ IN PERIOD T[J].
[4]   M←MARKETSHARE T
[5]   S←M×(⍴M)⍴INDSALES T
    ∇
```

4. Marketshare Function

```
    ∇ MS←MARKETSHARE T
[1]  ⍝ MS IS A MATRIX OF MARKET SHARES
[2]  ⍝ WITH MS[I;J]=MARKET SHARE OF
[3]  ⍝ ITH FIRM IN THE T[J]TH PERIOD.
[4]   T←,T
[5]   MS←(P[;T]*¯2)×(A[;T]*÷8)×D[;T]*0.25
[6]   MS←MS÷(⍴MS)⍴+/[1] MS
    ∇
```

5. Growth Function

```
    ∇ G←GROWTH T
[1]  ⍝ GROWTH IS A GOMPERTZ S-SHAPED CURVE
[2]   G←4000×0.2*0.9*T
    ∇
```

6. Seasonal Variation Function

```
    ∇ SV←SEASONAL T
[1]  ⍝ SEASONAL VARIATION IS A FUNCTION
[2]  ⍝ REPRESENTING A SEMI-ANNUAL
[3]  ⍝ INCREASE OF 15 PERCENT IN
[4]  ⍝ JANUARY AND JULY DUE TO
[5]  ⍝ CLEARANCE SALES.
[6]   SV←1+0.15×(1=12|T)∨7=12|T
    ∇
```

7. Merchandising Factor Function

```
    ∇ M←SALESINDEX T
[1]  ⍝ M[I]=INDEX FOR THE LEVEL OF
[2]  ⍝ INDUSTRY SALES DUE TO THE
[3]  ⍝ MERCHANDISING FACTOR FOR TIME T[I]
[4]   M←+/[1](P[;T]*¯2)×(A[;T]*÷8)×(D[;T]*
      0.25)
[5]   M←(M÷(1↑⍴P[;T])×(20*¯2)×(2500*3÷8))*
      2×1.05*-T
    ∇
```

8. Industry Sales Function

```
    ∇ I←INDSALES T
[1]  ⍝ I GIVES THE TOTAL INDUSTRY SALES
[2]  ⍝ FOR PERIOD T.
[3]   I←(SALESINDEX T)×(GROWTH T)×SEASONAL
      T
    ∇
```

B. The Data Bank and Discretionary Functions

The Data Bank consists of 3 arrays P,A, and D for Prices, Advertising Expenditures, and Distributional Expenditures. All Structural Functions depend upon the elements of the Data Bank, and use them as inputs. Each array consists of 2 rows and N columns corresponding to each of 2 competitors and N time periods. The Data Bank is generated using Discretionary Functions.

The Competitive Strategy Simulation involves 2 competitors each of whom must specify a marketing mix strategy. This involves choosing, for each time period, elements of the marketing mix; e.g., Prices, Advertising, and Distribution. Each Marketing Mix Strategy is represented by a unique Discretionary Function, and the major goal of the Simulation is the analysis of each strategy's relationship to other Marketing Mix Strategies. This is accomplished by the inclusion of a strategy's name, together with the names of other strategies it is to be compared with, in an appropriate 'LIST' provided to the Executive Processor. The Executive Processor, for each pair of strategies, calculates Data Bank elements, and calculates performance measures based upon the values in the Data Bank. Any number of alternative strategies can be specified and compared with other strategies as long as all strategy names appear in the 'LIST'. In this way each company can try to better its position by testing new strategies and modifying existing strategies.

Each Marketing Mix Strategy is represented by a Discretionary Function with the following syntax: *X←FNAME I* where *FNAME* is the function name, and *X* is a 3-element vector containing values for the marketing mix variables of price, advertising, and distribution for firm I in period T. The time period *T* is globally specified by the Executive Processor. The following functions only represent a sample for illustrative purposes.'

1. Non-Adaptive Strategies

S1A: Benchmark Marketing Mix

The model assumes a cost plus pricing structure where variable cost is constant at $10, and a benchmark value of P=$20 is considered reasonable. Current (pre-simulation) costs of advertising and distribution stand at $2500 each. These values are changed under alternative strategies to see what effect, if any, additional expenditures have on demand.

```
    ∇ X←S1A I
[1]   X← 20 2500 2500
    ∇
```

S1B: Market Skimming

With this strategy the firm tries to justify its high price by providing better distribution and spending more on advertising.

```
    ∇ X←S1B I
[1]   X← 25 3500 3500
    ∇
```

2. Time Dependent Strategy

S2D: Yearly Price Reductions

A price of $20 is set for the first 12 months, $18 for months 13 to 24, and $15 for months 25 to 36. This strategy attempts to capture consumers who are less price-conscious. prices are modified yearly instead of continuously.

```
    ∇ X←S2D I
[1]   →(T=1)/ONE
[2]   X←(20 18 15)[12 24 36 INTERVAL T],1.
      01×A[I;T-1],D[I;T-1]
[3]   →0
[4]  ONE:X←20,1.01×2ρ2500
    ∇
```

3. Competitively Adaptive Strategy

S3A: One-Period Lagged Imitation

This strategy serves as a benchmark for all of the adaptive strategies, and represents competition in a market in which there exists a price leader.

```
    ∇ X←S3A I
[1]   →ONE IF T=1
[2]   X←P[(2 1)[I];T-1],A[(2 1)[I];T-1],D[
      (2 1)[I];T-1]
[3]   →0
[4]  ONE:X← 20 2500 2500
    ∇
```

4. Sales Responsive Strategy

S4A: Marketing Mix as Function of Ratio of Percent Change in Sales[5]

When sales rise, price will be increased, advertising and distribution decreased by less than the percentage change in sales. The increases and decreases amount to 30% (the tolerance level) of the ratio of sales in the previous two periods. By changing the values in the vector *L* different percentages can be experimented with.

```
    ∇ X←S4A I;L;S1;S2
[1]   ⍝ SET TOLERANCE LEVEL FOR ADJUSTMENT
      PROCESS
[2]   L←3ρ0.3
[3]   S2←400
[4]   →(T= 1 2)/ONE,TWO
[5]   S2←(,SALES T-2)[I]
[6]  TWO:S1←(,SALES T-1)[I]
[7]   X←(P[I;T-1]×1+L[1]×¯1+S1÷S2),(A[I;T-
      1]×1+L[2]×¯1+S2÷S1),D[I;T-1]×1+L[3]×
      ¯1+S2÷S1
[8]   →0
[9]  ONE:X← 20 2500 2500
    ∇
```

5. Profit-Responsive Strategy

S5A: Marketing Mix as Function of Ratio of Percent Change in Profit

```
    ∇ X←S5A I;M
[1]   M← 4 3 ρ 0.96 1.02 1.02 0.98 1.01 1.
      01 1 1.02 1.02 1.02 1 1
[2]   →(T=1)/ONE
[3]   X←M[0.97999 1 1.02 INTERVAL PRATIO I
      ;]×P[I;T-1],A[I;T-1],D[I;T-1]
[4]   →0
[5]  ONE:X←(0.98×20),2ρ1.01×2500
    ∇
```

C. Executive Processor

The Executive Processor executes all functions necessary to perform a "Simulation". Each firm chooses a number of strategies and performance measures are calculated over a planning period of *N* months. Values are calculated for compounded cumulative net profit and final market share.

```
    ∇ EXEC S;T;J;X;I;P;A;D
[1]   ⍝ S IS A MATRIX OF STRATEGY NAMES,
        ONE PER ROW.
[2]   ⍝ J IS THE NUMBER OF STRATEGIES.
[3]    J←(ρS)[1]
[4]    T←(J,J,4)ρ0
[5]    I←1
[6]   ⍝ X IS A MATRIX OF UNIQUE PAIRS TAKEN
        FROM ⍳J.
[7]    X←⌽2 COMBIN⍳J
[8]    X←X,[1](⍳J)∘.+ 0 0
[9]   L: ⍝ FOR EACH PAIR OF STRATEGIES,
[10]  ⍝ CHECK IF STRATEGY PAIR IS VALID.
[11]   →L1 IF S[X[I;1];] STRATEGYCK S[X[I;2
        ];]
[12]  ⍝ CALCULATE MARKET MIX VARIABLES
[13]   S[X[I;1];] MARKETMIX S[X[I;2];]
[14]  ⍝ CALCULATE CUMULATIVE NET PROFITS
        AND FINAL MARKET SHARES
[15]   T[X[I;2];X[I;1];⍳4]←T[X[I;1];X[I;2];
        ⍳4]←(CNP PROFIT⍳N),,MARKETSHARE N
[16]  L1:→L IF(1↑ρX)≥I←I+1
[17]  ⍝-----------------------------------
[18]  ⍝ THE REST OF THE FUNCTION IS FOR
        OUTPUTING RESULTS
[19]  ⍝-----------------------------------
    ∇
```

The syntax specified for the Structural and Discretionary Functions provides great flexibility when designing an Executive Processor. The functions for Profit, Cost, Sales, Marketshare, and Salesindex all use a vector of time periods as input, and return a matrix as output, where each row corresponds to a competitor, and each column a time period. The functions for Growth, Seasonal Variation, and Industry Sales return a vector since the results are industry wide. To calculate any performance measure requires an APL statement constructed from combinations of Structural Functions. This is particularly useful when simulations using different market structures are compared.

V. Sensitivity Analysis: An Example

An example of how the conceptual framework provides for sensitivity analysis involves a detailed study of a particular pair of strategies. If competitor 1 uses strategy S1A and competitor 2 uses S5A, then a comparison is made between a non-adaptive strategy and a profit dependent strategy. Marketing instruments are calculated by the APL statement *'S1A' MARKETMIX 'S5A'* which produces the Data Bank variables as global variables. If one looks at profits over 36 periods (*PROFIT* ⍳36) and compares them to marketshare (*MARKETSHARE* ⍳36), one notices that competitor 2 enjoys high profits in the middle months, but declining profits and increasing marketshare in the later months. This could lead to an alternative strategy which changes after profits decline beyond a given point. The performance measures of cumulative compounded net profit, and final marketshare would not point out this behavior.

VI. Conclusion

Computer simulations are powerful tools for learning about the behavior of modeled systems. The conceptual framework developed in this paper provides a comprehensive methodology for programming simulation models with decreased programming effort and increased flexibility. It requires no special code, but does require strict adherence to rules for function specification and workspace organization.

Notes

1 See Kotler, Philip, "Competitive Strategies For New Product Marketing Over The Life Cycle," Management Science, Volume 12, Number 4, December, 1965, pages B-104 to B-119.

2 Kotler, Ibid., Page B-104.

3 This paper presents only a sample of Discretionary Functions.

4 An expanded version of this paper with all APL functions can be obtained from the author.

5 Strategy S4A which produced wild fluctuations in P,A, and D in Kotler's original paper has been corrected by the inclusion of an appropriate dampening factor.