An Algorithm for Calculating Indices in Fàa di Bruno's Formula

by

Eugene M. Klimko

Purdue University

Department of Statistics
Division of Mathematical Science
Mimeograph Series #277

April 1972

Abstract

This paper presents a fast algorithm for generating the non zero indices used in Fàa di Bruno's formula for the higher order derivatives of compositions of functions. The application of this formula to the solution of functional equations and to some problems in queueing theory are indicated.

Introduction

We consider real valued functions f and g of a real variable x whose nth order derivatives $f^{(n)}$, $g^{(n)}$ exist. An old formula due to Fàa di Bruno [1] gives the nth order derivative of the composite function h=fog expressed in terms of the derivatives of the functions f and g. The formula is given as

$$h^{(n)}(x) = \sum_{r=1}^{n} Y_{nr}\, f^{(r)}(g(x)), \tag{1.1}$$

where $h^{(n)}(0)$ is the nth derivative of h, $f^{(r)}$ is the r'th derivative of f and

$$Y_{nr} = \sum_{(j_1,\ldots,j_n)\in \mathcal{J}_{nr}} \frac{n!}{j_1!j_2!\ldots j_n!} \left(\frac{g'(x)}{1!}\right)^{j_1} \ldots \left(\frac{g^{(n)}(x)}{n!}\right)^{j_n}, \tag{1.2}$$

with $g^{n}(x)$ the nth derivative of g evaluated at x and

$$\mathcal{J}_{nr} = \{(j_1,\ldots,j_n): \sum_{i=1}^{n} j_i=r,\ \sum_{i=1}^{n} i j_i=n\}\ . \tag{1.3}$$

The major problem in using Fàa di Bruno's formula is the calculation of Y_{nr} in (1.2). Specifically, one must enumerate the terms occuring in the sum on the right hand side of (1.2). In the pre-computer era, this difficulty has limited the use of Fàa di Bruno's formula to calculating derivatives of small orders (at most 4 or 5). The purpose of this paper is to present a computer algorithm for rapidly calculating the coefficients Y_{nr} in (1.2).

The key idea of our approach to this calculation is to generate all vectors in the set

$$\mathcal{J}_n = \bigcup_{r=1}^{n} \mathcal{J}_{nr}\ . \tag{1.4}$$

The particular class $\mathcal{J}_{nr}$ to which a given vector belongs is then easily determined by computing the sum of its components. Moreover, it was discovered during the initial stages of this research that most of the components of a given vector $(j_1,\ldots,j_n)\varepsilon\mathcal{J}_n$ are zero. Therefore, it is advantageous to generate and use only the non zero components of each vector and the algorithm generates precisely these components.

In section 2, we describe the index generating algorithm in detail and give a Fortran program of it. We also introduce some notation which is used in section 3 to prove that the algorithm actually produces all of the vectors in $\mathcal{J}_n$. In section 4, we describe some applications of this algorithm. These include an application to queueing theory which also showed that in spite of the large number of terms occurring in the sum in (1.2), the procedure is numerically stable.

2. Index Generation Algorithm.

In this section, we give details of the algorithm for generating the vectors in the sets $\mathcal{J}_n$ described in the introduction. As each vector is generated, an index r designating the class $\mathcal{J}_{nr}$ to which the vector belongs is also calculated. This approach was necessary since there does not seem to be a simple method for generating the classes $\mathcal{J}_{nr}$. On the other hand, there is no disadvantage to this method since the entire set of coefficients Y_{nr} r=1,...,n may be simultaneously generated. Each vector $(j_1,\ldots,j_n)$ determines one term

$$\frac{n!}{j_1!\ldots j_n!}\left(\frac{g^{(1)}(x)}{1!}\right)^{j_1}\ldots\left(\frac{g^{n}(x)}{n!}\right)^{j_n} \tag{2.1}$$

and the index r tells to which partial sum representing Y_{nr} the term (2.1) should be added.

From the computational point of view, only the non zero components in the vector $(j_1,\ldots,j_n)$ are needed to evaluate (2.1). Therefore only these components are used in the algorithm. This approach is also advantageous in that it conserves storage space. Instead of the set $\mathcal{J}_n$, we shall deal with the collection $\mathcal{S}_n$ of elements

$$S = (\nu,\ r,\ I_\nu,\ J_\nu) \tag{2.2}$$

satisfying the conditions

$$\left.\begin{aligned} &1 \le \nu \le n \\ &I_\nu=\{i_k\}_{k=1}^{\nu} \\ &J_\nu=\{j_k\}_{k=1}^{\nu} \\ &r = \sum_{k=1}^{\nu} j_k \\ &n = \sum_{k=1}^{\nu} i_k j_k \\ &1 \le i_k \le n \\ &1 \le j_k \le n \end{aligned}\right\} \quad k=1,2,\ldots,\nu \tag{2.3}$$

The sets $\mathcal{S}_n$ and $\mathcal{J}_n$ may be mapped one to one onto each other by the relationship $\mathcal{V}$

$$(j_1',\ldots,j_n') = \mathcal{V}(\nu,\ r,\ I_\nu,\ J_\nu) \tag{2.4}$$

where $j'_{i_k}=j_k$ $k=1,\ldots,\nu$, $j'_i=0$ otherwise. The mapping $\mathcal{V}$ is clearly one to one and onto.

Since each vector generated by the algorithm may be represented as a transformation $\mathcal{U}$ applied to the previous element generated below, we define such a transformation on $\mathcal{S}_n$. Briefly, the algorithm may be described as beginning with the initial element $S_1 = (1,1,\{n\},\{1\}) \in \mathcal{S}_n$ and forming the successive powers

$$S_1,\ \mathcal{U}S_1,\ \mathcal{U}^2 S_1,\ \ldots,\ \mathcal{U}^N S_1 = S_0 \tag{2.5}$$

where $S_0=(1,n,\{1\},\{n\})$ and N is some fixed integer which is one less than the cardinality of the set $\mathcal{J}_n$.

The sequences $\{i_k\}_{k=1}^{\nu}$, $\{j_k\}_{k=1}^{\nu}$ are kept in stacks called I and J in the program listed in table I. In addition, the parameters ν, r and K are represented in the program by the symbols NU, R, K where

$$r = \sum_{1}^{\nu} j_k \ , \quad K=n-\sum_{k=1}^{\nu} i_k j_k \tag{2.6}$$

Whenever the algorithm completes the process of generating on element $\mathcal{U}$ S, K must be zero. The main steps in generating all elements of $\mathcal{J}_n$ are

A. initialize the stacks I_ν, J_ν and the parameters r, K, ν to $\nu=1$, $r=1$, $K=0$, $i_1=n$, $j_1=1$,

B. issue one vector,

C. test for termination condition: $i_1=1$, (2.7)

D. transform the quantities I_ν, J_ν, r, K, ν,

E. go back to B.

We now describe step D in detail. This step is the application of $\mathcal{U}$ to the element $S\in\mathcal{J}_n$ represented by the parameters ν, r, I_ν, J_ν. In this step, the stacks I, J are modified by one or more of the following operations.

a. delete the last element in the stacks,

b. modify the last element in the stacks, (2.8)

c. enter a new element in the stacks.

Each application of step D deletes at most two, enters at most two and deletes at most one element from the stacks. A modification of the last stack element is defined as setting $j_\nu=j_\nu-1$. Entry of a new stack element requires the

computation of values for $i_{\nu+1}$ and $j_{\nu+1}$. Entry is always done by pairs of elements (i*,j*) with i* entered into stack I and j* entered into stack J. Similarly deletion and modification always involves both stacks I and J; we shall sometimes refer to the stacks I and J collectively as the stack. Moreover, following each operation 2.8 a-c it is required that r and K be modified in such a way that (2.6) holds for the current status of the stacks I and J whether or not they are complete, that is, represent an element of $\mathscr{S}_n$. This requirement saves the summation which would otherwise be required to calculate r. The rules for modifying ν,r and K are

$$\begin{aligned} &\text{A. deletion: } r=r-j_\nu \quad K=K+i_\nu j_\nu,\ \nu=\nu-1,\\ &\text{B. modification: } r=r-1,\ K=K+j_\nu,\\ &\text{C. entry: } \nu=\nu+1,\ r=r+j_\nu \quad K=K-i_\nu j_\nu\ . \end{aligned} \tag{2.9}$$

Having defined the above rules, we may now describe the steps in carrying out part D of the algorithm

$$\begin{aligned} &\text{D.1. If } i_\nu=1 \text{ delete the last stack entry } (\nu=\nu-1),\\ &\text{D.2. Modify the entry at } \nu,\\ &\text{D.3. If } j_\nu \text{ is now 0, delete the entry at } \nu,\\ &\text{D.4. Set } i^*=i_\nu-1,\ j^*=[k/i^*],\\ &\text{D.5. If } j^*\neq 0, \text{ enter the pair } (i^*,j^*) \text{ into the stack}\\ &\quad (\nu=\nu+1,\ i_\nu=i^*,\ j_\nu=j^*),\\ &\text{D.6. If K is not 0, enter the pair (K, 1) into the stack}\\ &\quad (\nu=\nu+1,\ i_\nu=K,\ j_\nu=1) \end{aligned} \tag{2.10}$$

where [•] is the greatest integer function. The steps (2.10) define an application of the tranformation $\mathcal{U}$ to an element $S\varepsilon\mathscr{S}_n$ represented by the parameters ν r I_ν, J_ν. These steps will be translated into a set of operations on elements of $\mathscr{S}_n$ in the next section.

We conclude this section with the presentation of a Fortran subroutine of the algorithm. The input to the subroutine is N which is the value of the subscript n of $\mathcal{J}_n$ the desired class. The output vectors are issued by a call statement CALL PRNT. The output parameters are issued in the common block called I2. The subroutine PRNT disposes of the vector generated and is expected to leave the common block I2 intact.

```
                SUBRCUTINE INCEX
000001          CCMMCN/I2/N,R,NU,I(10),J(10)
000001          INTEGER R
          C
          C     INITIALIZE STACKS AND PARAMETERS
000001          I(1)=N
000002          J(1)=1
000003          R=1
000004          NU=1
000005          K=0
          C
          C     ISSUE CNE VECTOR
000006    10      CCNTINUE
000006          CALL PRNT
          C
          C     TERMINATION TEST
000007          IF(I(1).EQ.1)RETURN
          C
          C     BEGIN NEW VECTOR
000013    16      K=K+I(NU)
000015          R=R-1
          C
          C     MODIFY LAST STACK ENTRY
000016          J(NU)=J(NU)-1
000017          IS=I(NU)-1
000021          IF(IS.EQ.0)GOTO11
          C
          C     IF J(NU)=C, PURGE LAST ENTRY FROM STACK
000022          IF(J(NU).EQ.0)NU=NU-1
          C
000025    14      JS=K/IS
000030          IF(JS.EQ.0)GCTO15
          C
          C     MAKE NEW ENTRY IN THE STACK
000032          NU=NU+1
000033          J(NU)=JS
000035          I(NU)=IS
000036          K=K-IS*JS
000037          R=R+JS
          C
          C     IF STACK IS CCMPLETE GO TO ISSUE OUTPUT
000041          IF(K.EQ.0)GCTC10
000042    15      CONTINUE
          C
          C     MAKE A NEW ENTRY IN THE STACK
000042          NU=NU+1
000044          J(NU)=1
000046          I(NU)=K
000047          R=R+1
000050          K=C
000051          GCTO10
          C
          C     COME HERE TO PURGE LAST STACK ENTRY IF I(NU)=1
000051    11    K=K+J(NU)
000053          R=R-J(NU)
000055          NU=NU-1
000056          GCTO16
000056          END
```

Table 1

3. Proof of the Algorithm.

We now consider the proof that the algorithm generated in Section 2 generates all of the vectors in $\mathcal{J}_n$. Instead of the class $\mathcal{S}_n$ and the transformation $\mathcal{U}$, it will be more convenient to work with sets of vectors having n components rather than sets of the type $\mathcal{S}_n$ used in the previous section. We let $\mathcal{S}_n^*$ be the collection of all elements generated by the algorithm of Section 2, that is those given by

$$S_1, \mathcal{U}S_1, \mathcal{U}^2S_1, \ldots, \mathcal{U}^{n_0}S_1 \tag{3.1}$$

where S_1 is the element $(1,1,\{n\},\{1\})$ and $\mathcal{U}^{n_0}S_1$ is the termination element of the algorithm. We shall show here that the algorithm always terminates, that is the sequence (3.1) is finite.

Definition 3.1. Let $\mathcal{J}_n^*$ be the set of vectors $J^*=(j_1^*,\ldots,j_n^*)$ which correspond to some element in $\mathcal{S}_n^*$ by the relationship $\mathcal{V}$ defined by (2.4).

The transformation $\mathcal{U}$ on $\mathcal{S}_n$ induces a transformation $\mathcal{T}$ on the set $\mathcal{J}_n$ in the following way.

Definition 3.2. Let $J_1 \varepsilon \mathcal{J}_n$, $J_1 \neq (n,0,0,\ldots,0)$ and let S_1 correspond to J_1 under the mapping $\mathcal{V}$ then $\mathcal{T}J_1 = \mathcal{V}\mathcal{U}S_1$. If $J_1=(n,0,0,\ldots,0)$, then $\mathcal{T}J_1=J_1$.

Since the mapping $\mathcal{V}$ is one to one and onto, the transformation $\mathcal{T}$ is well defined. In fact, the pairs $(\mathcal{S}_n,\mathcal{U})$ and $(\mathcal{J}_n,\mathcal{T})$ are isomorphic under the mapping $\mathcal{V}$ except that $\mathcal{U}$ is not defined for the element S_0 for which $S_0=(n,0,0,\ldots,0)$. This however causes no difficulty.

Our goal becomes that of proving the following.

Theorem 3.1.

For any $n \geq 1$, $\mathcal{J}^*{}_n = \mathcal{J}_n$.

We postpone the proof until further results are established. However, we outline the idea of the proof. There is a natural ordering on the set $\mathcal{J}_n$ with the property that the initial vector used by the algorithm is the largest of all elements in the set $\mathcal{J}_n$ and the terminal vector is the smallest. The transformation $\mathcal{T}$ in turn generates successively smaller elements of $\mathcal{J}_n$ until it terminates. We shall now formalize this idea.

Definition 3.3.

Let $J_1=(j_{11},j_{12},\ldots,j_{1n})$, $J_2=(j_{21},j_{22},\ldots,j_{2n})$ be elements of $\mathcal{J}_n$. Then $J_1 \prec J_2$ if and only if $j_{1m} < j_{2m}$ where m is the largest integer such that $j_{1m} \neq j_{2m}$.

Definition 3.4.

Let $J_1, J_2 \in \mathcal{J}_n$. Then $J_1 \preceq J_2$ if and only if either $J_1 \prec J_2$ or $J_1 = J_2$.

We now establish some properties of the relationships $\prec$ and $\preceq$.

Lemma 3.1.

The relationships $\prec$ and $\preceq$ are transitive and form a trichotomy on $\mathcal{J}_n$, i.e. for any $J_1, J_2 \in \mathcal{J}_n$ either $J_1 \prec J_2$ or $J_1=J_2$ or $J_2 \prec J_1$. Moreover, for any $J \in \mathcal{J}_n$, $J_0 \preceq J \preceq J_1$ where $J_k=(j_{k1},\ldots,j_{kn})$ $k=0,1$ with $j_{01}=n$, $j_{0k}=0, k > 1$ and $j_{1n}=1$, $j_{1k}=0, k < n$.

The proof of this lemma is almost trivial and therefore is omitted.

In order to prove Theorem 3.1, we shall need a list of properties of the transformation $\mathcal{T}$ similar to the list of properties (2.10) of $\mathcal{U}$. We state them in the following.

Lemma 3.2.

The transformation $\mathcal{T}$ may be characterized by the following rules. Let $J_1=(j_{11},j_{12},\ldots,j_{1n})\in\mathcal{J}_n$. Then $\mathcal{T}\ J_1=(j_{21},j_{22},\ldots,j_{2n})$ where

(a) $j_{2s} = j_{1s}$ $\qquad s > t$

(b) $j_{2t} = j_{1t}-1$

(c) $K = n - \sum_{s=t}^{n} s\, j_{2s}$

(d) $j_{2(t-1)} = [K/(t-1)]$

(e) $i = n - \sum_{s=t-1}^{n} s\, j_{2s}$

(f) $j_{2i} = 1$ if $i > 0$

(g) $j_{2s} = 0$ if $1 \le s < t-1$, $s \ne i$

and t is the smallest of the integers $2,\ldots,n$ for which $j_{1t} > 0$.

This lemma is a direct translation of the properties D.1-D.6 of (2.10) into the space $\mathcal{J}_n$. Therefore we omit a proof. We now state the following.

Lemma 3.3.

For every $n=1,2,\ldots,\mathcal{J}_n^* \subset \mathcal{J}_n$

Proof.

We show that if $J=(j_1,\ldots,j_n)\varepsilon\mathcal{J}_n$, then $\mathcal{T}J\varepsilon\mathcal{J}_n$. We dispose of the trivial case by noting that if $J=(n,0,0,\ldots,0)$, $\mathcal{T}J=J\varepsilon\mathcal{J}_n$. Henceforth we assume that $J\neq(n,0,0,\ldots,0)$. Since $J\varepsilon\mathcal{J}_n$ we have

$$\begin{aligned} &j_i \geq 0 \qquad\qquad i=1,2,\ldots,n \\ &\sum_{i=1}^{n} i\, j_i = n. \end{aligned} \tag{3.3}$$

Let t be the smallest of the integers $2,\ldots,n$ such that $j_{1t} > 0$. Such a t exists because $J\neq(n,0,\ldots,0)$. Let $J'=(j_1',\ldots,j_n')=\mathcal{T}J$. Then $j_s'=j_s$ for $s > t$. Moreover, $j_t \geq 1$ and $j_t'=j_t-1 \geq 0$. Therefore $j_s' \geq 0$ for $s \geq t$. Since

$$\sum_{i=t}^{n} i\, j_i' = \sum_{i=t}^{n} i\, j_i - t, \tag{3.4}$$

we have

$$K=n - \sum_{i=t}^{n} i\, j_i' > 0. \tag{3.5}$$

It follows that $j_i' \geq 0$ for $i=2,\ldots,n$ since $j_s'=0$ or 1 if $s < t-1$. By (3.2), $j_1' \geq 0$. Clearly

$$\sum_{i=1}^{n} i\, j_i = n. \tag{3.6}$$

The lemma is proved.

We now demonstrate a key property of the transformation $\mathcal{T}$.

Lemma 3.4.

Let $J_1=(j_{11},\ldots,j_{1n})\in\mathcal{J}_n$. Then $\mathcal{T}J_1 \prec J_1$ unless $J_1=(n,0,0,\ldots,0)$.

Proof.

Let $J_2=(j_{21},\ldots,j_{2n}) = \mathcal{T}J_1$. Let t be the smallest of the integers $2,\ldots,n$ for which $j_{1t} > 0$. Then by (3.2a), $j_{2s}=j_{1s}$ for $s=t+1,\ldots,n$ and $j_{2t} < j_{1t}$. The lemma is proved.

A much stronger version of Lemma 3.4 is the following.

Lemma 3.5.

For any vector $J\in\mathcal{J}_n$ with $J\neq(n,0,0,\ldots,0)$, $\mathcal{T}J$ is the largest vector J' in $\mathcal{J}_n$ such that $J' \prec J$.

Proof.

We denote the components of vectors J by j_i $i=1,\ldots,n$ where J and j_i may have primes. Let $J''\in\mathcal{J}_n$ and $J'' \prec J$. Further let t be the largest integer for which $j''_t \neq j_t$, ie $j''_t < j_t$ since $J'' \prec J$. Finally, let u be the smallest of the integers $2,3,\ldots,n$ such that $j_u > 0$. We now argue the cases (i) $t=u$, (ii) $t > u$ and (iii) $t < u$. We first consider case (i).

If $t=u$, then by 3.2a, $j'_s = j_s$ for $s > t$ and it follows that $j''_s = j_s$ for $s > t$. Since $j''_t < j_t$, $j''_t \leq j_t-1=j'_t$. If either $j''_t < j'_t$, or $j''_t = j'_t, j''_{t-1} < j'_{t-1}$, then $J'' \prec J'$. If $j''_t = j'_t$, $j''_{t-1} \geq j'_{t-1}$, then we see that

$$K'' = n-\sum_{i=t}^{n} i\, j''_i = n-\sum_{i=t}^{n} i\, j'_t = K', \tag{3.7}$$

since $j''_i=j'_i$ for $i \geq t$. We must have $j''_{t-1} \leq [K/(t-1)]=j'_{t-1}$. If $t-1=1$, then $J''=J'$. If $t-1 > 1$ and $j''_{t-1}=j'_{t-1}$, then we let q be the remainder after division of K by t-1 and we note that for $q < s < t-1$, $j''_s = 0$ and $j''_q < 1$, for otherwise we would have

$$\sum_{i=q}^{n} i\, j''_i > n. \tag{3.8}$$

Since $j'_s = 0$ for $q < s < t-1$ by (3.2) and also by (3.2) $j'_q = 1$, it follows that $J'' \prec J'$.

We next consider the case $t > u$. Then $j'_s = j_s$ for $s > u$ and it follows that $j''_t < j_t = j'_t$ which proves that $J'' \prec J'$. The case $t < u$ cannot occur. We have $j_t > 0$ while j_u is the smallest of the integers $2,3,\ldots,n$ for which $j_u > 0$. Therefore t must be 1. However, if $j_s = j''_s$ for $s=2,\ldots,n$, then $J'' = J$. The Lemma is proved.

The preceding lemma is the key to proving theorem 3.1 to which we now turn.

Proof of Theorem 3.1.

We have seen that $\mathcal{J}_n^* \subset \mathcal{J}_n$. We now show that $\mathcal{J}_n^* = \mathcal{J}_n$. By means of the relationship $\prec$, we order the elements of $\mathcal{J}_n$ into a sequence $(n,0,0,\ldots,0)=J_1 \prec J_2 \prec \ldots \prec J_N=(0,0,\ldots,0,1)$ where N is the cardinality of $\mathcal{J}_n$. This is possible because $\preceq$ is a trichotomy on $\mathcal{J}_n$. Suppose that $J_{m^*} \in \mathcal{J}_n$ but $J_{m^*} \notin \mathcal{J}_n^*$. Then by Lemma 3.5 $J_{m^*} = \mathcal{T}J_{m^*+1}$. Therefore $J_{m^*+1} \notin \mathcal{J}_n^*$. By a finite induction, $J_s \notin \mathcal{J}_n^*$ for $s \geq m^*$, but $J_N \in \mathcal{J}_n$. The contradiction proves the theorem.

4. Applications.

Here we give some applications of Fàa di Bruno's formula. The basic application is that of finding the derivatives of the solution of functional equations of the form

$$g(x) = f(h(x)+g(x)) \tag{4.1}$$

where f and h are known g(x) is unknown and appropriate derivatives exist. We apply Fàa di Bruno's formula to (4.1) to obtain

$$g(X) = f(h(X)+g(X))$$

$$g^n(X) = \sum_{r=1}^{n} Y_{nr}\, f^{(r)}(h(X)+g(X)) \tag{4.2}$$

$$= Y_{n1}\, f^{(1)}(h(X)+g(X)) + \sum_{r=2}^{n} Y_{nr}\, f^{(r)}(h(X)+g(X))$$

where

$$Y_{nr} = \sum_{(j_1,\ldots,j_n)\in \mathcal{J}_{nr}} \frac{n!}{j_1!,\ldots,j_n!} \left(\frac{h'(X)+g'(X)}{1!}\right)^{j_1} \cdots \left(\frac{h^{(n)}(X)+g^{(n)}(X)}{n!}\right)^{j_n}. \tag{4.3}$$

The only element in $\mathcal{J}_{n1}$ is the vector $(0,0,\ldots,0,1)$. Therefore, $Y_{n1} = g^{(n)}(X)+h^{(n)}(X)$. For every $r > 1$, every vector $(j_1,\ldots,j_n)\in \mathcal{J}_{nr}$ has $j_n=0$ which means that the coefficients Y_{nr}, $r=2,\ldots,n$ depend on $g^{(1)}(X)\ldots g^{n-1}(X)$ and not on $g^{(n)}(X)$. This permits the establishment of the following recurrence relationship for the derivatives of g:

$$g^n(X)=h^{(n)}(X)f^{(1)}(h(X)+g(X)) + \frac{1}{[1-f^{(1)}(h(X)+g(X))]} \left[\sum_{r=2}^{n} Y_{nr} f^{(r)}(h(X)+g(X))\right] \tag{4.4}$$

$$g^{(1)}(X) = \frac{f^{(1)}(g(X)+h(X))h^{(1)}(X)}{1-f^{(1)}(h(X)+g(X))} \tag{4.4}$$

$$g(X) = f(h(X)+g(X)) \ .$$

The preceding idea may be applied to multiple compositions of functions with a corresponding increase in complexity of the relations (4.4).

This principle has been applied by Professor Marcel F. Neuts and the author [2] to the computation of the busy period moments of a single server queue with group arrivals. This computation involved a threefold application of Fàa di Bruno's formula to a functional equation of the type

$$\gamma(s) = h[s-\lambda+\lambda\theta[\log\gamma(s)]], \tag{4.5}$$

where h and θ are known functions and γ is unknown. We spare the reader the rather complex details of this application, but point out that some rather startling results were obtained. In [2], forty derivatives (moments) of γ were obtained with a moderate amount of computer time and it was observed that fifty moments were entirely possible. The surprising results of [2] were that despite the enormous number of terms (see table II) occurring in the summations of the type (4.3), no problems of round off error occurred. In fact, at least seven decimal places were obtained.

Table II

n	cardinality of $\mathcal{J}_n$
10	42
20	627
30	5604
40	37338
50	204226

We refer the reader to [2] for a wealth of information on computational organization and use of Fáa di Bruno's formula.

Finally, we present some results due to the referee who has written programs in the language SYMBAL which calculated the derivatives of h = f(g(x)) by means of Fáa di Bruno's formula and also by direct differentiation of h. Of interest was the timing information given in table III.

Table III Time in seconds		
Computation of	Fáa di Bruno	Direct differentation
h^{iii}	0.19	0.14
h^{iv}	0.38	0.34
h^{v}	0.63	0.74
h^{vi}	1.10	1.46
h^{vii}	1.73	2.70
h^{viii}	2.83	4.78
h^{ix}	5.50	8.19
h^{x}	7.50	13.72
h^{xi}	12.10	20.83

5. Acknowledgement.

The author would like to express his gratitude to Professor Marcel F. Neuts for his inspiration of this paper and also to the referee who tested the algorithm and obtained the timing data.

Bibliography

1. Fàa di Bruno, C. (1876) Théorie des Formes Binaires. Librairie Brero, Succr de P. Marietti, Turin, Italy.

2. Klimko, E. M. and Neuts, Marcel F. (1972) The Single Server Queue in Discrete Time-Numerical Analysis II. Naval Research Logistics Quarterly. (To appear 1973)