WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311

Calculating & Computing Applications

April 20, 1967

LOCI User Application Letter #3 p 1

The Application Letter for this month contains three items which can be extremely valuable in any program library. For statisticians, L46 & L49 are very useful. For the physical scientist or engineer, L48 is of incalculabel value. Incidentally, L48, written by Dr. Bonardi, contains an extremely elegant and instructive analysis.

L46-67 Poisson Distribution $P(K, \nearrow)$

L48-67 Solution for Second Order Differential Equation,
$y'' = F(x, y, y')$.

L49-67 Multiple Regression - 3 variables
$6 = b_o + b_1 x_1 + b_2 x_2$

If you would like copies of the programs, please return the bottom part of this letter. We shall also be happy to add anyone else to our mailing list, if you send in their names, addresses, etc.

Sincerely,

Ned Chang
Manager, LOCI Division

WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311



April 24, 1967

LOCI User Application Letter #3

L46-67
p 1

# Poisson Distribution

by

Caldwell Smith
Air Force Cambridge Research Laboratories

The Poisson Distribution is a discrete distribution used to describe certain processes, such as equipment failure and radioaction decay. The distribution is given by

$$P(k,\lambda) = e^{-\lambda} \left(\frac{\lambda^k}{ki}\right) \qquad (1)$$

In equation (1), k is the frequency and P (k, $\lambda$) is the probability associated with the frequency. The value $\lambda$ is usually taken to be the expected frequency.

The program is very simple to operate. A LOCI-2a is needed. The parameter $\lambda$ may be in the interval .0000001 to 100. Accuracy is to seven-places.

## OPERATING INSTRUCTIONS:

(1) Card in Reader, AUTO DISP up , "AUTO"
(2) PI to start
(3) Key in $\lambda$ , RUN , Read K = 0, RUN
(4) Read P (o,$\lambda$), RUN, Read K = 1, RUN
(5) Read P (1,$\lambda$), RUN, Read K = 2, RUN
Read P (k,$\lambda$), RUN
k < 100.

NOTE: Any time after step 3, $\sum_k P(k, \lambda)$ may be obtained by PO.

## EXAMPLE:

Suppose that the expected total equipment failure is 10 per week. We wish to tabulate the expected number of weeks where we have 0 failures, 4 failure . . . ,

| Number of Failures K | Expected Number of Weeks P(K,10) |
|---|---|
| 0 | .0000454 |
| 1 | .0004540 |
| 2 | .0022700 |
| ⋮ | ⋮ |
| 9 | .1251100 |
| 10 | .1251100 |
| 11 | .1137364 |

WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311



April 20, 1967



## Second Order Differential Equation

by
Dr. G. Fonda-Bonardi
Litton Systems

We have recently received a most interesting and useful program from Dr. Bonardi. This is a solution for a second order differential equation. The equation may be non-linear, and can be expressed as below:

$$y'' = F(x, y, y')$$

At this point, I quote from Dr. Bonari's letter of transmittal. "Since the program has several interesting features, I enclose a brief discussion of it and two sample runs which demonstrate how it is used and what results it produces. The examples are well known differential equations for which tabulated functions are readily available: this permits a check of the accuracy of the program and it is not intended to suggest that the program should be used in general to compute such functions when other subroutines are available (including looking them up in the tables). I use it, of course, to compute functions for which no tables can be found, as is normally the case in problems of plasma physics. I am sure that people working in other fields of physics and engineering have a similar experience: lack of published tabulations is the rule for differential equations actually met with in tackling practical problems. The enclosed program allows one to make up a tabulation of a function as needed, without having recourse to a big computer.

WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311



April 20, 1967



The program, on one card, takes care of the computation of any function y (with continuous derivatives) defined by

$$(1)\quad y'' = F(x, y, y')$$

where F is any algebraic function of the variables that can be normally handled by LOCI. In each case the user must tell the computer what his F (x, y, y') is, and he does so by coding a second card with his F exactly as it appears in equation (1): in other words, he programs the sequence of operations on x, y, and y' that results in y" according to equation (1).

The variables are taken from fixed memory positions that are always the same:

x is stored in MS0 S3

y is stored in MS4 S0

y' is stored in MS4 S2.

For example, supposing that the differential equation is

$$y'' = 1 - x + y',$$

(a rather trivial case), Card 2 would be coded with

1 - x + y'

| 1 | W→A | S3→W | - | MS | S2→W | + | RESTORE |
|---|---|---|---|---|---|---|---|
| 21 | 44 | 57 | 15 | 10 | 55 | 13 | 65 |

Then the user enters the initial values of the variables $x_o$, $y_o$, and $y'_o$ in the prescribed memory positions, with $y''_o$ in MS4 S3 and the constant

0.001 in MS4 S1, and starts with P0. The machine does the rest and prints a table of y, y' and y" versus x for increments of 0.01 in x. The accuracy (to six decimal places) is more than adequate for most practical purposes."

The program is written in a most efficient manner. It has additional features in the way of finding integrals of such as

$$\int f(y)\ d\ x$$

as well as inbedded functions such as

$$y'' = F\ (x,\ y,\ y',\ \sin(x)\ ).$$

WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311



April 20, 1967



## Operating Procedure for
## Second Order Differential Equation

Operation of this program is extremely simple. Since solution of any differential equation requires an understanding of the particular equation, as well as the method of solution, we suggest that the reader first peruse the analysis on the following pages. The minimal required hardware configuration is a LOCI-2ab. While the program is written for a printer, it is not absolutely necessary. We simply need to replace the WRITE commands at steps #02, 06, 08 and 10 by STOP's. Thus, instead of automatically printing x, y, y', and y", the values can be visually read. After each stop, the program resumes upon pushing RUN.

EXAMPLE

The program solves the D.E.:

$$y'' = F(x, y, y')$$

We can choose, as an example, the simple case

$$y'' = -y$$

Since the algorithm is on the first card, the second card simply calculates -y. It contains 5 steps:

| Step # | Command |
|---|---|
| 00 | MS |
| 01 | SO→A |
| 02 | A→W |
| 03 | + |
| 04 | RESTORE |

WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311



April 20, 1967

LOCI User Application Letter #2



To run the program:

(1) AUTO DISP down.

(2) Cards in Readers

(3) PRIME

(4) W--$S_3$ ($x_o = 0$)

(5) MS A--$S_0$ ($y_o = 0$)

(6) W--$S_3$ ($y''_o = 0$)

(7) 1 W--$S_2$ ($y'_o = 1$)

(8) . 0 0 1 W--$S_1$ ( x = .001)

(9) PO

With a printer, the program tabulates x, y, y', y". Without a printer, the program stops for each of the above. Push RUN to continue.

Some Results.

| | |
|---|---|
| x | .00 |
| y | .00 |
| y' | 1.00 |
| y" | .00 |
| x | .01 |
| y | .0099998353 |
| y' | .9999500019 |
| y" | -.0099998353 |

WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311



April 20, 1967



## Program For Second Order Differential Equation

by

Dr. G. Fonda-Bonardi

The behavior of many physical systems of interest in engineering and physics is described by second order differential equations. Some such differential equations can be solved in closed form, but the vast majority of them, due to the well known principle of Perversity of Nature, are of some non-linear form that cannot be handled analytically but can only be computed numerically.

The enclosed program tabulates the values of any function y defined by the general second order differential equation

(1) $y'' = F(x, y, y')$

in which F is any algebraic function of the variables x, y and y', and in many cases may be a function involving transcendental functions of the variables as well.

The program requires two cards: Card 1 contains the computing algorithm, Card 2 contains the equation (1) to be integrated, and is programmed by the user according to his needs. The program for Card 2 is written to yield y" as a function of x, y, and y' exactly as it appears in the defining equation (1). The only constraints on programming Card 2 are on the memory positions allocated to the variables:

x MS0 S3

y MS4 S0

y' MS4 S2

In addition, MS4 S1 contains the increment $\Delta x$ and MS4 S3 is used for y". Card 2 is entered at step -01 with MS in position 0, and must be programmed to end up with y" in the W register and MS in position 4. Memory positions MS0 S0, MS0 S1 and MS0 S2 are available for storing intermediate results of the computation and/or needed constants, and so are, if needed, all positions of MS8 and MS12. When the computed value of y" is in W and MS is on 4, the program is closed with RESTORE (65).

Iteration Algorithm

The computation is carried out by iteration of a Taylor expansion:

(2) $y_{n+1} = y_n + y'_n \Delta x + \frac{1}{2} y''_n \Delta x^2$

(3) $y'_{n+1} = y'_n + y''_n \Delta x + \frac{1}{2} y''_n \Delta x^2$

(4) $x_{n+1} = x_n + \Delta \bar{x}$

The initial values $x_0$, $y_0$, $y'_0$ and $y''_0$ are inserted in the memory before starting. At each iteration, equation (2) is computed in steps +50 to +65 of the program, and part of (3) in steps +66 to +74: at this point only $y'_n + y''_n \Delta x$ is available, and this is temporarily stored. Then equation (4) is computed, and Card 2 is entered to compute the new value of $y''_{n+1}$ using the new values of the variables just found. Since the value of $y'_{n+1}$ contains a small error due to the missing third term of equation (3), the new value of y' also contains a small error; however, it is close enough that it can be used to compute y" with negligible error:

(5) $y''_{n+1} = \frac{y''_{n+1} - y''_n}{\Delta x}$

thus equation (3) becomes:

(6) $y'_{n+1} = y'_n + y''_n \Delta x + \frac{1}{2} (y''_{n+1} - y''_n) \Delta x.$

Since both $y''_{n+1}$ and $y''_n$ are now available, the last term is added to $y'_{n+1}$ in Steps +24 to +36, and the correct value of $y''_{n+1}$ is computed again by re-entering Card 2, this time using the correct value of $y'_{n+1}$. At this point we have $x_{n+1}$, $y_{n+1}$, $y'_{n+1}$ and $y''_{n+1}$ and the process can be repeated for the next iteration.

Computation Errors

The errors involved in truncating the Taylor expansion after three terms are of the order of

$$1/6 \ \Delta x^3$$

except in pathological cases where the higher derivatives are extremely large. It does not appear advantageous to attempt an accuracy higher than what the LOCI can provide in computation. Since all multiplactions carried out in the LOCI may have an error in the last significant figures, the truncation error will not exceed the computation error if $\Delta x^3/6$ is of the order of $10^{-10}$. Hence a value $x = 0.001$ can be conveniently chosen as a good compromise between accuracy and speed of computation. Accordingly, the value 0.001 must be entered in MS4 S1 when setting up the initial values.

It is not convenient to print out every iteration, because printout is time-consuming and paper-accumulating. For most cases an adequate table of values can be constructed if the independent variable has increments of 0.01. Therefore, the program contains instructions to iterate ten time between printouts. I have found that this is quite satisfactory for most differential equations: but in special cases the user may wish to print out at finer or coarser intervals; this is done by changing steps +11 and +12 to reflect the desired number of iterations between printouts.

Initial Values

Initial values for x, y, y' and y" must be manually inserted in the memories before starting the program. Depending on the form of F (x, y, y') it is possible that no non-zero values can be computed starting from $y_o = y'_o = y''_o = 0$ (since x increases in any case, some F's can be started, but one must look into it). An example that does not start by itself for $P \geq 2$ ($y_o = y'_o = y''_o = 0$ at x = o) is

(7) $$y'' = -\left[\frac{y'}{x} + \left(1 - \frac{P^2}{x^2}\right) y\right].$$

An example that starts (at x = 1) is

(8) $$y'' = (y' - y + 1) \log x.$$

Equations like (7) must be handled carefully near x = 0 because the unavaoidable errors of computation have a large effect in the subsequent development of the function: the last two digits of a product, for example $P^2$, when divided by $x^2$ on the first iteration, have an effect that appears

on the fifth or sixth significant figure of the computed y", and therefore the program proceeds as if a different y" had been inserted as initial condition. Therefore the computed function is of the right kind but may really belong to a set of initial conditions that differs from the one actually entered. If such behavior cannot be tolerated, one must start with an $x>0$.

Program Check-out and Examples

The program can be best checked out by setting up some simple differential equation for which the solution is known, and comparing the results with tabulated values of the known function. Two such functions are presented as examples:

$y'' = -y$ (sines and cosines)

$y'' = -\left[\frac{I}{x} y' + \left(1 - \frac{p^2}{x^2}\right) y\right]$ (Bessel functions of order p)

Attached are the coding sheets for these two examples. Comparison of printed results with the published values for these functions shows that the computed values are accurate to at least six decimal places after four thousand iterations, which indicates a reasonably slow propagation of the truncation and computation errors. (Note: the computation for $J_1(x)$ was started at $x = 0.02$ to avoid unpleasantness near $x = 0$.) Of course, one would not normally use this program to compute simple functions, when other routines are available.

Special Features

The statement of the differential equation in Card 2 usually will occupy only a small portion of the card. The rest can be used to compute associated functions that may be of interest. For example, in one case I had a problem where the quantity I was after was defined by

$$2\pi \int_0^x xe^{-2y^2}\,dx.$$

where in turn y was defined by a complicated, Bessel-like differential equation.

Having y, it is quite easy to compute the value of the integral by accumulating in a suitable storage place, e.e. MSO SO, the sum of

$$2\pi\, e^{-2\,y^2}\, x\Delta x$$

at each iteration. Other problems may create similar requirements, for generating some other function of the variables being computed. The program has provisions for this in steps +42 and +43: these occur after all variables have been computed at each iteration, but before deciding for another iteration or a printout as the case may be. Step +41 to +43, as written, simply tell the machine to go ahead with the decision. However, by changing the address in +41 and +42 and W-PC (40) to W → XPC (41) in +43, one enters Card 2 and one can perform all additional desired operations. At the end of these there should be a sequence which looks up the number in DC (without changing it) and prints out the computed result when DC = 0, but skips printing if DC $\neq$ 0, thus providing a record of the computed result along with the others in the printed tabulation. In either case,

Card 1 is re-entered with 44 W-PC (24, 24, 40) at the end of the sequence in Card 2, and computation of the next iteration proceeds from there. Thus, assuming that the supplementary result is in W at the end of the computation, Card 2 should end with

70 24 24 40 11 24 24 40.

If more than one supplementary result is computed in this fashion, they should all be printed out before returning to step +44; the programming for this is obvious. Alternately, steps +4 to +10 may be rewritten to print out the supplementary results rather than the original functions.

It happens sometimes that the equation y" = F (x, y, y") contains a transcendental function of one of the variables that cannot be programmed as a combination of algebraic operations. Aside from the availability of subroutines for the computation of such functions, it is often possible to compute the desired function as the solution of an auxiliary second-order differential equation which is integrated step-by-step simultaneously with the integration of the main equation. Suppose, for example, that we have

(9) y" = F (x, y, y', cos wx)

set u = cos wx

then the auxiliary differential equation is

(10) $u'' = w^2 u$

If the MS8 group of memories is assigned to u, $\Delta$ x, u' and u", then the new value of $u_{n+1} = \cos wx_{n+1}$ can be computed at each iteration simply by letting steps +50 to +75 operate on the values stored in MS8 rather than in MS4, under control of Card 2, in conjunction with a program stating eq. (10). The same course can be done with any trascendental function that has a simple generating differential equation. This algorithm results in appreciable saving of time, because the transcendental function is obtained with only on computing per iteration of the main program (which has to be performed anyway), and saving of card space, because the algorithm is already in Card 1, and all that is needed in Card 2 is the auxiliary differential equation (which may be quite short, e.g. equation (10) ) and a command to and from Card 1, plus a suitable shift of MS and back. Care should be exercised in taking advantage of the peculiar properties of the nested STORE-STORE-RESTORE-RESTORE commands if they are used in performing these operations. Unless familiar with them, substitute direct transfers by W-PC (40) and W-xPC (41) if familiar with them, use the available steps +77, +72, +79 to straighten things out. (Incidentally, if the transcendental function happens to be a sine or a cosine, there is a quicker way to do it, even though this has been used as an example above).

WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311



April 20, 1967



Computation Time

The computation time depends, of course, on the complexity of the function F (x, y, y'). A program of average complexity such as the Bessel function requires about 45 seconds between printouts. Long and slow programs are best left running overnight.

WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311



April 24, 1967

LOCI User Application Letter #3



## Multiple Regression - 3 Variables

Bill Smith
Wang Laboratories, Inc.

As an initial exercise on the LOCI, our new staff member wrote this regression analysis program. The program can be used to find a linear relationship between 3 variables. The value $r^2$ (square of correlation) yields a measure of the relation between the variables.

Suppose we have a sequence of data points =

$$\begin{array}{l} Y_1, X_{11}, X21 \\ Y_2, X_{12}, X_{22} \\ \vdots \quad \vdots \quad \vdots \\ Y_3, X_{1n}, X_{2n}. \end{array}$$

The regression procedure employed here finds coefficients $b_o$, $b_1$ and $b_2$ for the following equation:

$$Y = b_o + b_1 x + b_2 x \qquad (1)$$

The coefficients are determined so as to minimize the sums of the squares of the deviations,

$$\sum_i [Y(X_i) - Y_i]^2.$$

Five cards are used. The first card simply clears out the registers. The second card is used to enter all the data. It computes and stores the following quantities:

$$\Sigma X_1 X_2,\ \Sigma Y,\ \Sigma X_1,\ \Sigma X_2,\ \Sigma X_2 Y,\ \Sigma X_1 Y,\ \Sigma Y^2,\ \Sigma X^2,\ \Sigma X_2^2,\ N.$$

Cards #3, #4, and #5 compute the results; $r^2$, $b_o$, $b_1$, and $b_2$.

Equipment Needed: LOCI-2a

WANG LABORATORIES, INC. 836 NORTH ST., TEWKSBURY, MASS. 01876, TEL. (617) 851-7311



April 24, 1967

LOCI User Application Letter #3



OPERATING PROCEDURE:

1. AUTO DISP down, "AUTO".
2. Load Card #1
3. Push $P_o$.
4. Load Card #2.
5. Push $P_o$.
6. Key in Y, RUN.
7. Key in $X_1$, RUN.
8. Key in $X_2$, RUN.

** Repeat steps 6, 7, 8 for all data points.

9. Load Card #3.
10. Push $P_o$.
11. Load Card #4.
12. Push $P_o$.
13. Load Card #5.
14. Push $P_o$.
15. Read $r^2$, RUN.
16. Read $b_o$, RUN.
17. Read $b_1$, RUN.
18. Read $b_2$.

Follow steps #1 - #5 at left.

3 RUN 8 RUN 1 RUN

10 RUN 20 RUN 4 ± RUN

4 RUN 7 RUN 9 RUN

5 RUN 1 RUN 12 RUN

Follow steps 9 - 18 at left.

$r^2$ = .7402

$b_o$ = -1.3309

$b_1$ = .5913

$b_2$ = .3355

EXAMPLE:

| Y | 3 | 10 | 4 | 5 |
|---|---|---|---|---|
| $X_1$ | 8 | 20 | 7 | 1 |
| $X_2$ | 1 | -4 | 9 | 12 |