VECTOR

APL97 at Toronto

The Journal of the British APL Association

ISSN 0955-1433
www.vector.org.uk
Vol.14 No.2 October 1997

A Specialist Group of the British Computer Society

Contributions

All contributions to VECTOR may be sent to the Journal Editor at the address on the inside back cover. Letters and articles are welcome on any topic of interest to the APL community. These do not need to be limited to APL themes, nor must they be supportive of the language. Articles should be accompanied by as much visual material as possible (b/w or colour prints welcome). Unless otherwise specified, each item will be considered for publication as a personal statement by the author. The Editor accepts no responsibility for the contents of sustaining members' news, or advertising.

Please supply as much material as possible in machine-readable form, ideally as a simple ASCII text file on an IBM PC compatible diskette (any format). APL code can be accepted as camera-ready copy, in workspaces from I-APL, APL+Win, IBM APL2/PC or Dyalog APL/W, or in documents from Windows Write (use the APL2741 TrueType font, available free from Vector Production), and MS Word.

Except where indicated, items in VECTOR may be freely reprinted with appropriate acknowledgement. Please inform the Editor of your intention to re-use material from VECTOR.

Membership Rates 1997-98

Category	Fee	Vectors	Passes
UK Private	£12	1	1
Overseas Private	£14	1	1
(Supplement for Airmail, not needed for Europe)	£4		
UK Corporate Membership	£100	10	5
Overseas Corporate	£135	10	
Sustaining	£430	10	5
Non-voting Member (Student, OAP, unemployed)	£6	1	1

The membership year normally runs from 1st May to 30th April. Applications for membership should be made to the Administrator using the form on the inside back page of VECTOR. Passes are required for entry to some association events, and for voting at the Annual General Meeting. Applications for student membership will be accepted on a recommendation from the course supervisor. Overseas membership rates cover VECTOR surface mail, and may be paid in sterling, or by Visa, Mastercard or JCB, at the prevailing exchange rate.

Corporate membership is offered to organisations where APL is in professional use. Corporate members receive 10 copies of VECTOR, and are offered group attendance at association meetings. A contact person must be identified for all communications.

Sustaining membership is offered to companies trading in APL products; this is seen as a method of promoting the growth of APL interest and activity. As well as receiving public acknowledgement for their sponsorship, sustaining members receive bulk copies of VECTOR, and are offered news listings in each issue.

Advertising

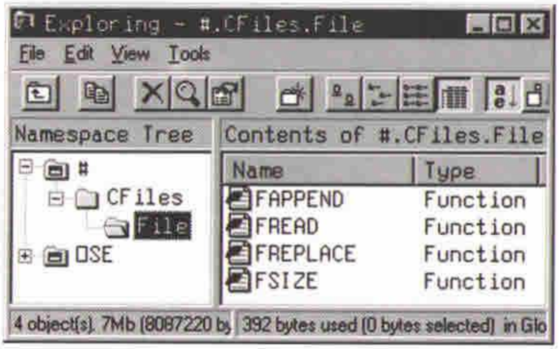
Advertisements in VECTOR should be submitted in typeset camera-ready format (A4 or A5) with a 20mm blank border after reduction. Illustrations should be photographs (b/w or colour prints) or line drawings. Rates (excl VAT) are £250 per full page, £125 for half-page or less (there is a £75 surcharge per page if spot colour is required).

Deadlines for bookings and copy are given under the Quick Reference Diary. Advertisements should be booked with, and sent to: Gill Smith, Brook House, Gilling East, YORK YO6 4JJ. Tel: 01439-788385 Email: apl385@compuserve.com

Contents

dyalog APL

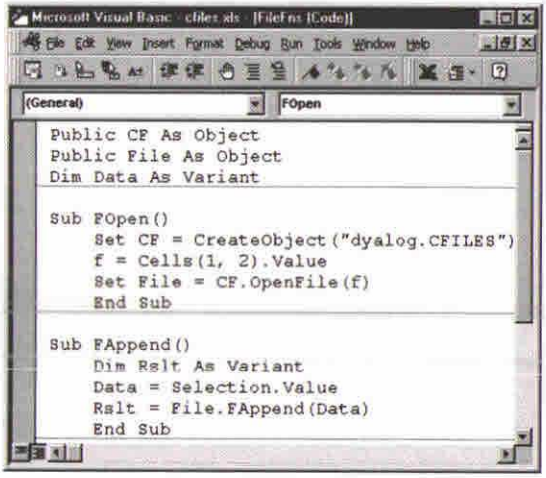
The Definitive APL for Windows™


Exploring - #.CFiles.File
File Edit View Tools
Namespace Tree
Contents of #.CFiles.File
#
CFiles
File
DSE
4 object(s). 7Mb (8087220 by
392 bytes used (0 bytes selected) in Glo

Name	Type
FAPPEND	Function
FREAD	Function
FREPLACE	Function
FSIZE	Function

Have you ever wanted to call your APL functions from Excel, Visual Basic, or C++ ?

Well now you can!


Microsoft Visual Basic - cfiles.xls - [FileFns (Code)]
File Edit View Insert Format Debug Run Tools Window Help
(General)
FOpen

```
Public CF As Object
Public File As Object
Dim Data As Variant

Sub FOpen()
    Set CF = CreateObject("dyalog.CFILES")
    f = Cells(1, 2).Value
    Set File = CF.OpenFile(f)
    End Sub

Sub FAppend()
    Dim Rslt As Variant
    Data = Selection.Value
    Rslt = File.FAppend(Data)
    End Sub
```

That's why Dyalog APL/W remains the **professional** choice. For further information, contact Dyadic or your local distributor today.

Dyadic Systems Limited, Riverside View, Basing Road, Old Basing, Basingstoke, Hants. RG24 7AL, United Kingdom.
Tel:+44 1256 811125 Fax:+44 1256 811130 Email: sales@dyadic.com

MICROSOFT® WINDOWS™ COMPATIBLE

Editorial

by Duncan Pearson

This issue of Vector is coming to you unfortunately three months late for which I apologise. We are sending out 14.2 and 14.3 together. Due to the pressure of work I will no longer be continuing as editor but the main strength of Vector, the Vector Working Group, will continue to produce the magazine. It is planned that a different member of the group will take responsibility for the editorial content of each issue and that the administrative and production responsibility will lie with Gill Smith. Adrian and Gill have been producing Vector now for over ten years and their contribution to the promulgation and support of APL cannot be over-estimated.

This issue contains the report of APL97 in Toronto which I rated very highly. The presentations that I attended were interesting and thought-provoking and the organisation of the conference was good. Unfortunately only Adrian and I from the Vector team went to the conference and so we only have the two reports. However some of the presenters have kindly written up their presentations as articles for Vector and these are included.

Contact Point for all Vector Material

Following the meeting of the Vector working group on February 7th, it was agreed to leave the post of Editor temporarily vacant, and to have the working group operate as an editorial board, with a 'guest editor' (either from the group or invited by them) responsible on an issue by issue basis. Please direct all correspondence to:

Vector Administration, c/o Gill Smith
Brook House, Gilling East
YORK YO6 4JJ

apl385@compuserve.com

It will then be distributed to the appropriate person on the working group. Many thanks.

J Release 3.05...

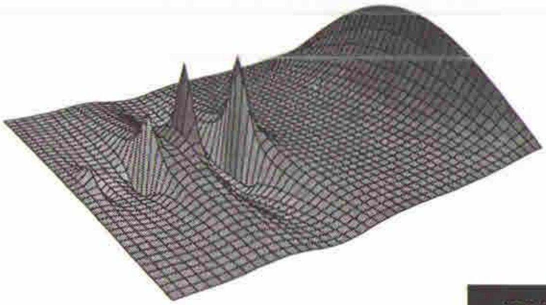
Jsoftware is an executable notation that forms the basis of a high-level general-purpose programming system.

J is based on simple and consistent rules that work directly on arrays of any size or dimension. The concise syntax and interactive environment make J ideal for applications such as statistics, financial and actuarial analysis, modeling, simulation and educational use.

J Release 3.05 adds standard regular expression pattern matching, and arbitrary-precision rational numbers. Automation support has been extended to allow Java programmers to use the full power of J in building applications and applets.

Real part of gamma function in range [(-3.5, -i), (4.5, i)]:

```
gamma=. ! @ <:
real=. {. @ +.
x=. steps _3.5 4.5 40
y=. steps _1 1 40
z=. real gamma x j./ y
dat=. _3 >. 12 <. z
'surface;viewpoint 1.5 _1 0.5' plot dat
```



Produced by:
Iverson Software Inc.
Distributed by:
Strand Software Inc.
19235 Covington Court
Shorewood, Minnesota
USA 55331
Tel (612) 470-7345
Fax (612) 470-9202
info@jsoftware.com

www.jsoftware.com

Quick Reference Diary 1998–1999

Date	Venue	Event
May 22nd	RSS, London	Vendor Forum and AGM
July 27-31	Rome	APL98 — see below
August 10-14 1999	Scranton, Penn.	APL99 — see next issue

Dates for Future Issues of VECTOR

	Vol.14 No.3	Vol.14 No.4	Vol.15 No.1
Copy date	-	10th April	3rd July
Ad booking	-	17th April	10th July
Ad Copy	-	24th April	19th July
Distribution	March 98	May 98	August 98

Vector Back Numbers

Back numbers of Vector are available from:

British APL Association,
c/o Gill Smith,
Brook House, Gilling East,
YORK YO6 4JJ

Price in UK: £10 per complete volume (4 issues); £12 (overseas); £16 (airmail) including postage.

APL98 — ROMA

About the Event

The APL98 conference will take place from 27th to 31st July 1998 at the Faculty of Economics of the University of Rome "Tor Vergata". This Faculty offers excellent conference facilities like a four hundred plus seats main conference room, several two hundred plus secondary conference rooms, computer room facilities, plenty of space both inside and outside and excellent parking facilities.

The conference is devoted to present the state of the art in the development and applications of array processing languages, with particular attention on computational environments and marketplace languages such as APL, J, Mathematica, Fortran90, Maple, Gauss, and others.

The conference topics focus on (but are not limited to):

- **languages and environments** — state of art, future prospectives, relation among languages
- **computer science** — human-computer interaction, object oriented programming, parallel architectures,parallelism and concurrency, networking
- **applications** — discrete mathematics, new computing paradigms (neural nets, genetic programming), finance, economics and social science, insurance and actuarial mathematics, statistics and operational research, simulation
- **education** — teaching APL(s), teaching with APL(s)

For more information, please contact:

Antonio Annibali, Faculty of Economy, University of L'Aquila
tel: (+39)-862-432401
fax: (+39)-862-432403
tel: (+39)-6-88327266 (home)

Paolo Di Chio, Faculty of Economy, University of L'Aquila
tel: (+39)-862-432428
fax: (+39)-862-432403
tel: (+39)-6-66153929 (home)
e-mail: mc0307@mclink.it

or send an e-mail to apl98@poeco.utovrm.it

Call for Papers

The landscape of computing is continually changing. New achievements bring new possibilities to the users and pose new challenges to both practitioners and theoreticians. On new frontiers of computing are such issues as distributed computing, location-independence, remote programming, interaction and smart agents.

Are APL, J and in general array processing languages (APLs) still able to live up to these challenges? Are APLers brave enough to come out of their niches and once again take their place on the leading edge of computer science? Consider how APL on its inception set the agenda for modern computing: interpretation, standard set of primitives designed for machine independence, inter process communication, parallelism, typed I/O. In the following years developers of APLs have come out with a wealth of enrichments of the original environment: full screen editors, object-oriented extensions, new control flow primitives, interfaces with other languages, environments and operating systems. The time has come to promote a new and central role for array processing languages in the new territories of end-user computing through their expressiveness, of INTERNET computing through the possibility of encapsulating data and programs, of high performance computing through their native management of parallelism.

Contributions are sought which will emphasize how array processing languages are a significant response to the new exigencies, how they allow the rapid development of significant applications both in classical and in new fields of use, how they provide adequate settings for users to develop their own applications.

Topics of interest include, but are not limited to:

1. **State of the art of APLs:**
   Present situation and future directions

2. **Computer Science:**
   Human Computer Interaction
   Object-Oriented Programming
   Parallel Architectures
   Parallelism and Concurrency
   Distribution
   Meta-level Programming
   INTERNET Computing

3. **Discrete Mathematics, Algorithms and new computing paradigms:**
   Neural Networks
   Genetic Algorithms
   DNA computing

4. **Applications:**
   Finance and Financial Maths
   Economics and Social Sciences
   Insurance and Actuarial Maths
   Statistics and Operational Research
   Image Processing
   Simulation of physical, biological and social phenomena
   Others

5. **Education:**
   Teaching APLs
   Teaching with APLs

Types of Contributions

Several types of contributions are welcome, namely:

Papers: 30/45 minutes scientific communication

Tutorials: 60/90 minutes knowledge dissemination (single or multiple sessions)

Workshops: 90 minutes technical hands-on (single or multiple sessions)

Poster/panel sessions

Birds-of-a-feather sessions

For each of them the following time schedule is provided:

Papers

February 7th: 1 page abstract, indicating the main topic and the subcategory of the contribution and the time requested (30/45 minutes)

February 28th: first acceptance
April 15th: first draft
May 15th: acceptance notification
June 15th: final version

Tutorials

March 1st: 2 pages abstract, indicating the topic/category of the tutorial, the time requested (60/90 minutes), if single or multiple session and the availability to possibly replicate the tutorial

April 15th: acceptance notification
June 15th: final version

Workshops

March 1st: 2 pages abstract, indicating the topic/category of the workshop, if single or multiple session, hardware and software requirements and the availability to possibly replicate the workshop

April 15th: acceptance notification
June 15th:final version

Panel-Posters

March 1st: 1 page abstract
May 1st: acceptance notification

Submission

Abstracts should be submitted on plain ASCII file and sent by e-mail (preferred), fax, or ordinary mail at the following addresses:

E-mail: mc0307@mclink.it
apl98_abstract@poeco.utovrm.it

Fax:(+39)-862-432403 (Attn. Prof. Antonio Annibali)

Ordinary mail:
Paolo Di Chio
Faculty of Economics, University of L'Aquila
Via Assergi, 6
67100 L'Aquila, Italy

Proceedings

Contributions will be published in the Conference Proceedings.

CORRESPONDENCE

Accounts Correction

From: Nicholas Small (Hon. Treas.) 3rd September 1997

It has been drawn to my attention that there is an error in the accounts summary as presented to the AGM and published in Vector 14.1. The figure for <Total payments> for 1996/97 (R&P) should read 22000 (don't ask me where 20149 came from!).

It is perhaps also worth mentioning that the amounts written off should appear on a separate line below net assets. They do not form part of the sum to calculate net assets; rather, they are for use in the reconciliation identity:

```
new_assets = (old_assets - written_off) + (receipts - payments)
```

Re: Year 2000

From: M.E. Martin 2nd October 1997

As I have been working on the Year 2000 problem for the last five months I have noted Bob Brown's letter (*Vector* 14.1 p. 8) and at first dismissed it, but on closer examination I thought there is a good reason to reply. If you can't bear the suspense please skip to the last paragraph.

I am currently working for a Life Insurance/Pensions company with APL systems dating back to 1984. Most were written by actuaries, pension administrators and apprentice APL programmers. As you can imagine dates are very important and used in many calculations. You would also expect that these programs must already cope with the next century; well some do, by using a three-digit year, e.g. 101 means 2001!

If you think about it you can represent a date in at least a dozen formats and that is just the numeric ones! Consequently we have some 64 functions just converting from one representation to another and back again. We have several programs to convert to and from serial (often called Julian) dates, some include the 400 leap year rule and a couple even the 4000 year rule! None of them comes up with 'standard' Julian dates (1/1/1900=1), and all get the 2000 leap year wrong! Most treat AD 1900 wrongly as a leap year (as does MS Excel v.4a), although this is unlikely to cause problems by now. Then we have lots of rate tables mostly keyed on a two-digit year, all references to ⎕*TS* must be examined, etc.

To complicate matters The Management in their wisdom has decreed that screens, printouts and file layouts (linking to non-APL systems) are NOT to be changed. Thus we are to internally 'window' the dates, i.e. any year <30 is presumed to be >2000. It may sound easy, but is 12/8/38 a date of birth or a retirement date? How do the date-handling functions tell the difference?

Examining all the APL code would take 16 man-years. By eliminating all the dead systems and code and concentrating on business critical systems we can do it in 3 man-years. It is too late to re-write the systems, and they would not let us anyway.

Bob Brown is right — it should never have happened — but this is not a perfect world and we the consultants have to tediously patch things up to last another few years.

Here is the proof. If Bob Brown examines his functions he will find that they are unaware of 2000 being a leap year!!!

M.E. Martin
General Software
22 Russell Road
Northolt
Middx UB5 4QS

Year 2000 — Why the Consternation?

From: Anne Wilson August 1997

Bob Brown's letter on this subject (*Vector* 14.1) shows a somewhat myopic view, looking at the problem from the point of one application maintained by one person. But the world is bigger than that, and *many mickles make a muckle* — a large number of little problems create a big problem.

Some problems will be external to the applications. The clock on my Compaq thinks that the date following 31/12/1999 is 4/1/1980 — one wonders why it is not 1/1/1980? If I then set it to the correct date most things are as expected. However, at least one accessory loses the ability to put a correct creation date on files. I can reset the date, but what would happen if the computer were controlling machinery?

In many commercial businesses there will be hundreds, if not thousands, of applications — and they all need testing and checking. Much of this code will have been written many years ago by programmers long since departed from the programming arena of that company — whereabouts in the programs does date processing occur — how many places does it need to be changed — what functions does it perform? A single programmer still maintaining his own code should find it easy to identify and correct, but who knows the workings of the mind of a long-departed programmer; who knows where to look for dates in code whose very purposes are lost in the mists of time, with many users and many alterations?

Files may not contain sufficient digits to hold years as 4 digits. To change the file layouts for many files, perhaps with many different date fields, can create a large implementation problem — imagine the potential size of the problem if it is for an insurance company or bank! And every program will probably need to be altered, even if only by recompilation. As this may originate from old Cobol programs, when storage was an expensive consideration, there may be thousands of references to 6-digit picture fields — how many need to be altered?

Bob describes his method of adding.the extra two digits for dates; this makes assumptions about which hundred-year time slot is needed. Such assumptions require knowledge of the system, and may well precipitate more alterations at some later date. Does his system cope correctly with the Queen Mother's date of birth in August 1900?

A decade ago many commercial programs had a life expectancy of 7 or 8 years; even databases were changed with amazing frequency. I can smugly say that all the systems I had control over hold 4-digit years; but I cannot say the same for the hundreds of programs in which I have been involved, but did not control. Way back in 1971 there were panics over the decimalization of money. What will the next panic problem be?

Anne D. Wilson
12 Thorny Hills
Kendal
Cumbria LA9 7AL

Heron's Rule & Integer-Area Triangles

From: William R. Jones 30 January 1998

This note offers a follow-up to Eugene McDonnell's *Heron's Rule & Integer-Area Triangles* (VECTOR 12.3 pp133-142) and Roger Hui's *Linear Recurrences and Matrix Powers* (VECTOR 12.4 pp113-115). It offers a simpler, fast solution to a problem they treat.

Mr. McDonnell's exploration of a certain class of integer-area triangles leads him to the sequence 1, 2, 7, 26, 97, 362,... with recurrence relation `A(n) = 4*A(n-1) - A(n-2)`. His J interpretation of the recurrence relation employs recursion and is intolerably slow. Subsequently he describes interesting applications of a generating function, Taylor's series, and partial fraction decomposition, which result in rapid generation of terms of the sequence, with the added benefit of an edifying excursion with some less well known J primitives. Enlightening as that development is, it's rather heavy machinery if the goal is to generate terms of the sequence speedily.

Mr. Hui interprets the recurrence relation as a matrix multiplication. In this interpretation generating terms of the sequence requires computing powers of a matrix, and Hui shows and compares three methods for accomplishing that task. This leads to very fast generation of terms of the sequence.

This sequence, and any which have linear recurrence relations, can be generated with an elementary approach which is transparent and quite fast. At any stage in generating this sequence we use the last two terms to generate the next term, append that to the sequence, and repeat this process. Generating the next term is simply taking an inner product of the vector `_1 4` and the last two terms. Repetition of the process is accomplished by function iteration — power conjunction (`^:`) in J. The verb s implements this method.

```
   s=. 3 : 'y. , _1 4 +/ .* _2{. y.'

   s 1 2              NB. A few example runs
1 2 7

   s 1 2 7
1 2 7 26

   s^:8 [ 1 2         NB. The first 10 terms
1 2 7 26 97 362 1351 5042 18817 70226
```

```
   {: s^:28 [ 1 2     NB. The 30th term
1.92952e16
```

This term is computed in less than an eyeblink on a modest machine:

```
   6!:2 '{: s^:28 [ 1 2'
0.09
```

William R. Jones (jonesw@lafcol.lafayette.edu)
Lafayette College

A Little Hook for Understanding

From: Nick Cox (N.J.Cox@durham.ac.uk) 4 September 1997

We all know how appropriate names or phrases can help us to grasp elements of APL. When starting out, the meaning of *f/* can be learned and remembered more easily by focusing on 'insert' or 'reduce', each of which conveys much about that basic construct.

In J, `@` is described in ISI literature as 'atop', but I find it easier to think of it as either 'applied to' or 'after', which I think I learned from the writings of Eugene McDonnell and Ken Smillie, respectively.

A common remark in learning J is that forks require a big mental effort and hooks a bigger one — after which you see them everywhere. In grappling with monadic hooks I find help from tiny phrases: mentally insert 'their own' or 'its own' between the verbs of the hook.

The monadic hook (`f g`) acting on `y` is equivalent to (`y f g y`).

`pr =. % +/` gives proportions.

`pr 1 2 3 4` gives 0.1 0.2 0.3 0.4

Think of this as 'divide by their own sum', or if you are truly thinking arrays and seeing the vector 1 2 3 4 as a whole, 'divide by its own sum'.

`- +/ % #` is a hook, consisting of minus `-` and the average or mean `+/ % #`, which seems to be everybody's first example of a fork.

Think of this as 'minus its own mean'. That is then a short step from the statistician's jargon 'deviation from the mean'.

Interesting Problem...

From: Ajay Askoolum 22nd July 1997

Given a chess board (8×8 cells), place exactly 16 dots, one in any given cell such that:

1. each row
2. each column
3. each diagonal

is either empty or has exactly 2 dots in it.

This can be judged by APL solution + execution time. (The solution is not unique ... but how many solutions are there?)

[An available solution has 3 lines & executes in 0.5 seconds on a 33Mhz 486]

Enumerating and Generating Combinations

From: Norman Thomson 12 September 1997

I should like to comment on the functions for enumerating and generating combinations given by Alan Wilson in his article on the National Lottery (see *Vector* Vol.14 No.1 pp.51-57). Under the heading "Which Combination?" a function *COMBNUMBER* is given which maps a combination *c* of *n* items out of *m* into a unique integer. This function can be given much more succinctly as

```
      ∇Z←c COMBNUMBER m;T
[1]   Z←1+((⍴c)!m)-1++/(⍳⍴T)!T←m-⌽c
      ∇
```

The basis for this is that positive integers can be represented uniquely by vectors of a predetermined length in such a way that the integer given by a vector `cv` standing for "combination vector" is `+/(⍳⍴cv)!cv`. For example the value of the integer represented by the combination vector `1 2 4 5` is `+/(⍳4)!1 2 4 5 = 11`. The uniqueness of this representation requires that the minimum values in the successive vector positions are `⍳cv` in index origin 0. Notice that a combination vector is not itself a combination.

The inverse function which computes the combination vector (`cv`) corresponding to a given integer is obtained in the following way. If `4=⍴cv`, the `cv` corresponding to 11 is built up from the right by first finding the largest *n* for which `4!n` is less than 11. This value is 5 since `4!6` is 15. Subtract `4!5` from 11 to give 6 and repeat for `3!n`. The largest *n* is 4 since `3!5 = 10`, so subtract `3!4 = 4` to give 2. Continuing in this way `cv` is developed as `1 2 4 5`. A function which implements the above algorithm is:

```
     ∇Z←n CI i;K
[1]  ⍝ Z is ith combination vector of length n
[2]   →(0≠i)/L0 ⋄ →0 Z←¯1+⍳n   ⍝ first stopping condition/action
[3]  L0:→(n>1)/L1 ⋄ →0 Z←i     ⍝ second stopping condition/action
[4]  L1:K←n                    ⍝ start recursive section
[5]  L2:→(i≥n!K←K+1)/L2        ⍝ loop until K too large
[6]  Z←((n-1)CI i-n!K-1),K-1   ⍝ recurse to find lower order digits
     ∇
```

Using this it is now simple to represent Mr. Wilson's *NUMBERCOMB* with slightly different parametrization as

```
     ∇Z←nm NUMBERCOMB i
[1]  Z←⌽(2⊃nm)-(↑nm)CI(!/nm)-i  ⍝ ith combination of n out of m
     ∇
```

These functions were given in my paper *Some Combinatoric Algorithms in APL* at APL75 in Pisa, and the foundations for using the combination vector representation of the integers in order to obtain orderly listings of combinations is well set out in the references given there, namely Algorithic *Combinatorics* by Shimon Even (Collier Macmillan, 1973) and *Applied Combinatorial Mathematics* ed. E.F. Beckenbeck (Wiley, 1964) — see the opening chapter "The Machine Tools of Combinatorics" by D.H. Lehmer.

Mr. Wilson observes that calculations pertinent to e.g. the National Lottery rapidly hit the phenomenon of combinatorial explosion, and a degree of pragmatism must be used in order to avoid workspace overflow. A significant advantage of representing combinations by combination vectors is that it is easy to obtain estimates of, say, the distribution of combination totals by Monte Carlo methods. Thus a reasonable short cut to full enumeration of such totals is obtained by taking say 1000 random numbers between 1 and 13,983,816, viz:

```
   +/¨(⊂6 49)NUMBERCOMB¨1000?13983816
```

British APL Association News

Future Issues of Vector

Issue 14.4 (April 1998) will be distributed at APL98 in Rome. This issue will be produced in May to ensure APL98 advance coverage.

Issue 15.1 (July 1998) will be ready in August, including coverage of APL98.

Issue 15.2 (October 1998) onwards will be prepared on schedule.

British APL Association 1998/1999 Committee

The members of the Committee are unpaid officials of the Association. However, reasonable expenses incurred in the execution of Association business are reimbursed.

One of the key objectives of the Annual General Meeting is to ensure that all posts are filled. The AGM attempts to fill every post for the duration of the forthcoming year. Any official of the Association can be elected to serve for a continuous maximum period of 3 years, although all posts are eligible to be filled by a new appointment.

One of the pressing issues at this AGM is the appointment of an Editor for Vector, the journal of the Association. In addition to control of the editor's budget, the Editor also has use of a portable PC.

The British APL Association is seen as the foremost international association and its publication Vector is regarded as the premier publication of its kind. Vector is also the only regular publication for APL news.

The Association relies on volunteers for existence and exists purely to promote the overall interests of the APL community. It requires the participation of new members; this not only ensures a fresh, and more effective, approach but also allows the existing volunteers to take a sabbatical.

If you need advance information on Committee roles and responsibilities, please contact any member of the existing committee — contact details are on the inside of the back cover of Vector.

The current committee urges you to attend the AGM and to volunteer to serve on the new committee. Your initiative is critical to the continued success of the British APL Association in serving the APL community.

APL Product Guide

compiled by Gill Smith

VECTOR's exclusive APL Product Guide aims to provide readers with useful information about sources of APL hardware, software and services. We welcome any comments readers may have on its usefulness and any suggestions for improvements.

We reserve the right to edit material supplied for reasons of space or to ensure a fair market coverage. The listings are not restricted to UK companies and international suppliers are welcome to take advantage of these pages.

For convenience to readers, the product list has been divided into the following groups ('poa' indicates 'price on application'):

- Complete APL Systems (Hardware & Software)
- APL Interpreters
- APL-based Packages
- APL Consultancy
- Other Products
- Overseas Associations
- Vendor Addresses
- World Wide Web and FTP Sites

Every effort has been made to avoid errors in these listings but no responsibility can be taken by the working group for mistakes or omissions.

We also welcome information on APL clubs and groups throughout the world.

Your listing here is absolutely free, will be updated on request, and is also carried on the Vector web site, with a hotlink to your own site. It is the most complete and most used APL address book in the world. Please help us keep it up to date!

All contributions and updates to the APL Product Guide should be sent to Gill Smith, at Brook House, Gilling East, York, YO6 4JJ. Tel: 01439-788385, Email: apl385@compuserve.com

COMPLETE APL SYSTEMS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Dyadic	IBM RS/6000 MD320	11,736	APL POWERstation (Greyscale) 27.5 MIPS, 7.4 Mflops RISC Processor 8Mb RAM, 120Mb Disk 19" 1280x1024 Greyscale Graph Display AIX, OSF Motif, Dyalog APL (1-user)
	IBM RS/6000 MD320	13,817	APL POWERstation (Colour) 27.5 MIPS, 7.4 Mflops RISC Processor 8Mb RAM, 120Mb Disk 16" 1280x1024 Colour Graphics Display AIX, OSF Motif, Dyalog APL (1-user)
	IBM RS/6000 MD320	22,656	Advanced APL POWERstation 27.5 MIPS, 7.4 Mflops RISC Processor 16Mb RAM, 320Mb Disk, 150Mb Tape 16" 1280x1024 Colour Graphics Display AIX, OSF Motif, Dyalog APL (1-user)
	IBM RS/6000 MD520	37,114	APL POWERsystem (8-users) 27.5 MIPS, 7.4 Mflops RISC Processor 16Mb RAM, 320Mb Disk, 150Mb Tape CD-ROM Drive, 16 Ports AIX, Dyalog APL (2-8 user licence)
	IBM RS/6000 MD530	72,054	APL POWERsystem (16-users) 34.5 MIPS, 10.9 Mflops RISC Processor 32Mb RAM, 1.34Gb Disk, 2.3Gb Tape CD-ROM Drive, 16 Ports AIX, Dyalog APL (8+ user licence)
	IBM RS/6000 MD540	122,842	APL POWERsystem (32-users) 41 MIPS, 13 Mflops RISC Processor 64Mb RAM, 1.7Gb Disk, 2.3Gb Tape CD-ROM Drive, 32 Ports AIX, Dyalog APL (8+ user licence)
Optima	IBM Compatible	poa	Complete networked or stand-alone solutions including configuration installation, maintenance and commissioning.

APL INTERPRETERS

COMPANY	PRODUCT	PRICES(£)	DETAILS
APL Software	APL*Plus/PC Release 10	450	STSC's APL for IBM PCs & compatibles.Upgrades from earlier releases also available.
	Run-time	poa	Closed version of APL*Plus/PC which prevents user exposure to APL
	APL*Plus II	1,395	All the features of mainframe APL*Plus for your 386PC!
	Run-time	poa	
	Dyalog APL	1000-10,000	2nd generation APL for Unix systems
	APL2/PC	poa	IBM's APL 2 for the PC.
Beautiful Systems	Dyalog APL/W for Windows	poa	US Distributor of Dyalog APL products from Dyadic.
	Dyalog APL for Unix	poa	See Dyadic listing for product details.
The Bloomsbury Software Company	APL+PC Version 11	260	Upgrade to version 11 gives free runtime (£120 from any version)
	APL+Win v1.8	1350	A 32-bit Windows-hosted interpreter that runs under all Windows platforms including Windows 95. Note: Any user purchasing APL+Win during 1996 will receive free updates to Vn 1.8 and Vn 2.0 (user to pay carriage)
	Upgrade to Version 1.8	540	From earlier versions of APL+Win. Free update to Version 2.0 (user to pay carriage)
	Migration to APL+Win	620	from APL*PLUS II versions 4/5. Free update to Version 2.0 (user to pay carriage)
		750	from earlier versions of APL*PLUS II
	APL+DOS	1300	APL*PLUS II DOS is renamed to APL+DOS.
	Migration to APL+DOS	620 / 390	from APL*PLUS/PC or APL*PLUS II

	APL*PLUS II for UNIX	poa	APL2000's 2nd generation APL for all major Sparc and Risc Unix workstations.
	APL*PLUS VMS	poa	2nd generation APL for DEC VAX computers under VMS.
	APL*PLUS Mainframe	poa	Enhances VS APL with many high performance, high productivity features. For VM/CMS and MVS/TSO offers simple upgrade from VS APL.
Dinosoft Oy	Dyalog APL/W for Windows	poa	Finnish distributor of Dyalog APL products.
	Dyalog APL for Unix	poa	See Dyadic's listing for product details.
Dyadic	Dyalog APL for DOS/386	995	Second generation APL for DOS.Runs in 32-bit mode, supports very large workspaces. Unique "window-based" APL Development Environment and Screen Manager. Requires 386/486 based PC or PS/2, at least 2Mb RAM, EGA or VGA, DOS 3.3 or later.
	Dyalog APL/W for Windows	995	As above, plus object-based GUI development tools. Requires Windows 3.0 or later.
	Dyalog APL for Unix	995-12,000	Second generation APL for Unix systems. Available for Altos, Apollo, Bull, Dec, HP, IBM 6150, IBM RS/6000, Masscomp, Pyramid, NCR, Sun and Unisys machines, and for PCs and PC/2s running Xenix or AIX. Oracle interface available for IBM, Sun and Xenix versions.
IAC/Human Interfaces			
	I-APL/Mac	13	Macintosh version of I-APL
I-APL Ltd	I-APL/PC or clones	8	ISO conforming interpreter. Supplied only with manual (see 'Other Products' for accompanying books).
	I-APL/BBC Master	8	As above
	I-APL/Archimedes	8	As above
	Strand Software Inc		Strand Software Inc has the sole selling rights to Iverson Software Inc products. I-APL stocks a few of these (mainly APL/WIN and the personal J products and books), but is no longer an agent.
IBM APL Products	TryAPL2	free	APL2 for educational or demonstration use. Write, fax or Email to APL Products; specify disk size desired.
	APL2 PC (US Version)	$630	Product No. 5799-PGG. PRPQ Number RJ0411. Order from 1-800-IBM-CALL
	APL2 PC (European Version)	£348	Product No. 5604-260. Part number 38F1753. From all IBM dealers, including MicroAPL.
	APL2 for OS/2 Entry Edition	$185	Part No 89G1556.
	APL2 for OS/2 Advanced Edition	$650	Part No 89G1697. Contains all facilities of the Entry Edition plus: DB2 interface; co-operative processing TCP/IP interface; tools for writing APs; TIME facility
	APL2 for Sun Solaris	$1500	Product No. 5648-065.
	APL2 for AIX 6000	poa	Product No. 5765-012.
	APL2 Version 2	poa	Product No. 5688-228. Full APL2 system for S/370 and S/390
	APL2 Application Envt Vn2	poa	Product No. 5688-229. Runtime environment for APL2 packages
Insight Systems	APL*PLUS/PC	poa	APL systems marketed and supported ...
	Dyalog APL	poa	from: Dyadic, Manugistics, IBM
	APL2	poa	under: Windows, OS2 and Unix
Iverson Software Inc.	J on the Web online registration ...		
	J Educational Edition	$50	
	J Standard Edition	$150	
	J Professional	$325	
	Books and accessories (discounts for reg users)		
	J Dictionary	$50	
	J User Manual	$50	

	J Phrases	$40	
	J Primer	$40	
	Set of the above 4 books	$160	
	Concrete Math	$40	
	Fractals, Visualization & J	$50	
	Exploring Math	$50	
	J User Conference Proceedings	$35	
	Mugs, T-shirts, Mousepads	$10 each	
	APL\WIN	$70	For 386/PC under Windows 3.1
J Austria	J	poa	Distributor for Austria and Switzerland
	Dyalog APL	poa	Distributor
	Causeway Products	poa	Distributor
	Structural Analysis Software	poa	Complete package by IG Zenkner&Handel to perform structural analysis/engineering calculations. Also suitable for dynamic problems, e.g. earthquake simulation.
Lescasse Consulting	APL+PC	poa	Lescasse Consulting is the exclusive APL2000 distributor in France and also
	APL+Unix	poa	distributes in Switzerland and Belgium. Call for price quotes.
	APL+DOS	poa	
	APL+Win	poa	
	Dyalog APL/W	poa	French distributor for Dyalog
MasterWork Software	Manugistics Products and ISI	poa	New Zealand distributor
MicroAPL	APL.68000 Level I	2000	First generation APL with numerous enhancements. Multi-user version (Unix, Mirage, MCS).
	APL.68000 Level II	2500	Second generation APL. Nested arrays, user defined operators, selective specification etc. Multi-user version (Unix, Mirage, MCS)
	APL.68000/X	1500-6000	Second-generation APL. Nested arrays, user defined operators, selective specification, etc. Multi-user AIX version with full OSF/Motif support.
	APL.68000 Level I Mac, ST, Amiga	87	First generation APL. Single user, full windowing interface, software floating point support.
	Mac, Amiga	260	First generation APL. Single user, full windowing interface, hardware floating point.
	APL.68000 Level II ST	170	Second generation APL. Full windowing interface, software floating point support.
	Amiga	260	Second generation APL. Full windowing interface.Hardware and software floating point support.
	Mac	520	Second generation APL. Full windowing interface.Hardware and software floating point support.
	APL*PLUS Rel 10 APL*PLUS II V 4.0	450 1395	
Oasis Systems	Dyalog APL	poa	Dyadic Systems
	APL*PLUS	poa	Manugistics
	APL.68000	poa	MicroAPL Ltd
	APL2	poa	IBM
Optima	Dyalog APL/W	995	Fully fledged Windows development environment.
RE Time Tracker Oy	APL+PC (APL*PLUS/PC)	poa	Complete APL+ and Statgraphics product range and links to various 3rd party products.

	APL+DOS (APL*PLUS II)		
	APL+Win (APL*PLUS III), APL+Link		
	APL+UNIX		
	APL*PLUS Sharefile		
Soliton Associates	SHARP APL for MVS	poa	for IBM MVS mainframes
	SHARP APL for Unix	poa	for IBM RS/6000 and Sun SPARC
Strand Software	Canada		
	All APL*PLUS Products	poa	All APL*PLUS products including upgrades and educational.
	Dyadic and ISI products	poa	
	USA		
	Dyadic and ISI products	poa	

APL PACKAGES

COMPANY	PRODUCT	PRICES(£)	DETAILS
ADAPTA Software	MPS - Master Production Schedulingpos		
	FBS - Forecasting and Budgeting System		
	DRP - Distribution Requirements Planning		
Adaptable Systems	FLAIR	poa	Finite loader and interactive rescheduler. Customisable full-function scheduling system. (Available outside Australia by special arrangement only.)
Adaytum Software	Adaytum Planning	poa	Full-featured Budgeting and Financial Planning system for medium to large enterprises.
The APL Group	Qualedi	$1500-4000	Electronic Data Interchange (EDI) translation software for the PC, with strict compliance checking.
APL Software Ltd (mainframe)	RDS	poa	Relation Data Base System
	IPLS	poa	Project Management System
	REGGPAK	poa	Regression Analysis Package
(microcomputer)	POWERTOOLS	295	Assembler written replacement function for commonly used CPU-consuming APL functions, includes a Forms Processor.
	REGGPAK	poa	Regression Analysis Package
	RDS	990	Relational Database System
Beautiful Systems	ASF_FILE	$399	Dyalog APL/W auxiliary processor for access to APL*PLUS/PC APL component files (*.ASF).
	NAT_FILE	$299	Dyalog APL/W auxiliary processor which emulates the APL*PLUS/PC quad-N native file subsystem for access to the DOS file system.
	DBF_FILE	$299	Dyalog APL/W auxiliary processor for efficient block mode access to dBASE format files. Designed to get large amounts of data in and out of dBASE. Not suited for random access to small amounts of data (it does not handle keys).
	SF_READ	poa	Dyalog APL/W functions to read APL*PLUS data objects of any type or structure from *.SF style component files created by APL*PLUS II or III.
The Bloomsbury Software Company (for VSAPL)	Enhancements & Sharefile	poa	Component files, quad-functions & nested arrays for VSAPL under VM/CMS & MVS/TSO
	Compiler	poa	The First APL compiler!

(for APL2)	Sharefile/AP	poa	STSC's shared access component file system for APL2. Comparable to all APL*PLUS file systems: multi-user storage of APL2 arrays with efficient disk usage.
Causeway	*CausewayPro* for Dyalog/W	400/$600	Causeway application development platform for Dyalog APL/W.
	RainPro Business Graphics	250	The ultimate graphics toolkit for the APL developer. Adds 3D charting capability, Web publishing and clipboard support to the shareware product. Charts can be included in *NewLeaf* reports. Functionally compatible across Dyalog/W and APL+Win.
	NewLeaf for Dyalog and +Win	400	Frame-based reporting tool with comprehensive table-generation and text-flow support. Offers multiple master-page capability, bitmap wrap-around and on-screen preview with pan and zoom. Fully supported on Dyalog/W and APL+Win (1.8 and above)
Cinerea AB	ORCHART	250	Organization chart package for IBM APL2/PC. Full & heavily commented source code included - free integration into other applications. NB: ASCII output with line-drawing (semi-graphic) characters for boxes.
CODEWORK	HELM	poa	Decision Support system for top management. Handles large multi-dimensional tables, data analysis, EIS presentations; generates HTML and Latex output. Platforms: DOS, APL+II, Windows 3.1/95 for Dyalog APL, LAN support. Ideal for APL customisation, more than 100 installed.
H.M.W.	4XTRA	poa	Front-end Foreign Exchange dealing / pos keeping
	Arbitrage	poa	Arbitrage modelling
	Basket	poa	Basket currency modelling
	Menu-Bar	poa	pull-down menu for APL*PLUS/PC
HRH Systems	APL Utilities	poa	Software to transfer workspaces between APL*PLUS and Sharp, and between APL*PLUS and I-APL. Software to import IBM .ATF files to APL*PLUS.
	APL*PLUS Utilities		Public domain software, unlock locked fns, a user-friendly alternative to locking, fns of mathematical physics, menus, and others.
IAC/Human Interfaces	SPARKS	poa	Educational simulation of electric circuit (for Apple Mac.)
	EPIDEMIC	poa	Educational simulation of spreading infection (for Apple Mac.)
	COINS	poa	Educational simulation (KS3) of coin-tossing experiment with simple stats (for Apple Mac.)
	FIBONA	poa	Educational simulation of Fibonacci's rabbits (for Apple Mac.)
I-APL Ltd	Educational workspaces	5	PC format disks with the examples from: Thomson, Espinasse (Kits 1-4), Kromberg, Jizba & FinnAPL. All the examples to save your fingers!
IBM APL Products	A Graphical Statistical System	$250	for DOS, Product Number 5764-009
	(AGSS)	$500	for Workstations (OS/2, Aix, Solaris), Product Number 6764-092
		$2500	for CMS, Product Number 5764-011
INFOSTROY	APL*PLUS/Xbase Interface (II/386 Version 2)	$198	Complete package written in C. Comparable with the data, index & memo files of FoxPro, dBASE, & Clipper. Multi-user support. No DBMS license required.
	(DLL Version 1)	$198	The same in a DLL form! Gives your Windows applications all advantages of DLLs.
Insight Systems	IUTILS/XP	20-95	Cross-platform utility library including simple OS calls (DIR, COPY, DEL, RENAME) and DATE functions. For APL*PLUS II, APL2 and Dyalog APL under Windows, OS/2 and Unix.
	ASI	95	APL Spreadsheet Interface. "Device-independent" spreadsheet driver supporting Excel, 123 and Quattro-Pro for Dyalog APL/W
	WinCom	95	Asynchronous comms package for Dyalog APL/W
	S2D,22D,X2X	poa	Advanced APL syntax analysis and conversion packages from Sharp and APL2 to Dyalog, and between any two APLs

	SQAPL Client	poa	Interface from APL*PLUS II, APL2 and Dyalog (Windows, OS/2 or Unix) to most SQL databases over most networks.
	SQAPL Server	poa	Makes APL*PLUS II, APL2 or Dyalog APL (Unix) available as SequeLink servers. Can be called from SQAPL clients or other applications such as Excel, C++, Smalltalk, Visual Basic.
JAD Software	JAD SMS	150-500	Software management system for APL*PLUS II based on hierarchical databases; includes full-screen interface and stand-alone functions. Price depends on number of users.
Lescasse Consulting			
	APL+Win Monthly Training Program	$600	Download 50+ page document about APL+ programming each month. You also get one or more workspaces full of re-usable APL code and sometimes additional files or products.
	Advanced Windows Programming ...	$95	200-page book plus companion disk on interfacing APL and Delphi. Contains full coverage of Delphi-2, +Win and Dyalog.
	DLL parser for +Win	$250	Parse any Visual Basic DLL declaration file into a set of quadNA definitions. Turn constants and structures into APL variables. Available for APL+Win and Dyalog/W.
	Delphi Forms Translator	$195	Design forms with Delphi and turn them automatically into APL programs which recreate the same form (+Win and Dyalog/W).
	APL+Link Pro	poa	ODBC Interface for APL+Win
	SQAPL Pro	poa	ODBC Interface for Dyalog APL/W
	RainPro	poa	Highly customisable 2D and 3D publication graphics for APL+Win and Dyalog APL/W
	NewLeaf	poa	Page layout and printing tools for APL+Win and Dyalog
	GraphX and ChartFX	poa	High-quality business graphics for APL+Win
	Formula One and Dyalog APL	$95	100-page book + companion disk on how to use the Formula One VBX with Dyalog APL/W
Lingo Allegro	Internet Server AP	poa	An internet server for Dyalog APL/W. Visit www.lingo.com for a live demo.
RE Time Tracker Oy	UIT/W	poa	Comprehensive high-level Windows User Interface library for APL+Win and +II v 5.1. Comprehensive spreadsheets, replicated fields, special field types, etc. 16 and 32 bit versions available.
	AJGRAPH	poa	Graphpak-compatible 2D graphics package for +Win and +DOS. Includes multi-window support, print and metafile support. No DLLs required.
	ECCO PRO with APL	poa	Leading group and personal information management system with comprehensive customising. Supplied with sample +Win workspace to interface to ECCO databases via DDE.
	NEWT TCP/IP SDK with APL	poa	Lead TCP/IP SDK with interfaces to all protocols. Supplied on 3 CD ROMS together with a sample +Win workspace.
	DB+	poa	Database interface for APL+DOS under Windows. Allows combining character-based APL applications with ODBC-compliant databases such as Oracle and SQL-server..
Soliton Associates	LOGOS	poa	Application Development Environment
	MAILBOX	poa	Electronic Mail
	VIEWPOINT	poa	Report generator with interfaces to DB2 and MVS data
Warwick University	BATS	250	Menu driven system for time series analysis and forecasting using Bayesian Dynamic modelling. Price is reduced to £35 for academic institutions.
	FAB	free	Training program for the above.
Zark	APL Tutor (PC)	$299	APL computer-based training. Available for APL*PLUS PC & APL*PLUS II. Demo disk $10.
	APL Tutor (MF)	$5000	Mainframe version.
	Zark ACE	$99	APL continuing education. APL tutor news and hotline phone support.

	APL Advanced Techniques....	$59.95	488pp. book, (ISBN 0-9619067-07) including 2-disk set of utility functions (APL*PLUS PC format).
	Communications	$200 pc, $500 mf	Move workspaces or files between APL environments.

APL CONSULTANCY AND DEVELOPMENT

COMPANY	PRODUCT	PRICES(£)	DETAILS
Adfee	Consultancy	poa	Development, maintenance, conversion, migration, documentation, of APL products in all APL environments
Ajay Askoolum	Consultancy	poa	APL+Win development and migration of actuarial, financial, mathematical applications.
Andrews	Consultancy	poa	APL programming and analysis, Year-2000 legacy systems, algorithms, tree-processing.
APL Solutions Inc	Consultancy	poa	APL systems design, development, maintenance, documentation, testing and training. Providing APL solutions since 1989.
AUSCAN Software	Consultancy	poa	APL software development, training
Bloomsbury Software	Consultancy	300-750+VAT	
Camacho	Consultancy	poa	Manuals; feasibility reports and estimates; analysis and programming; APL and MS Windows applications; Sharp, ISI APL, APL*PLUS, APL2/PC and other APLs spoken. Fixed price systems a speciality
Ray Cannon	Consultancy	poa	APL, C, Assembler, Windows, Graphics: PC and mainframe
Causeway	Consultancy and Training	poa	On-site training for Causeway, RainPro and NewLeaf. Customisation and enhancement to meet local needs. Code review and pre-implementation check of Causeway applications.
Paul Chapman	Consultancy	250-500	24-hour programmer: APL, Smalltalk, C; Windows front end design a speciality.
CODEWORK	Consultancy	poa	Development, maintenance, migration, documentation of APL applications. Speciality: Info systems for top executives, Internet applications.
Dinosoft Oy	Consultancy	poa	Specialised in very large databases.
Dyadic	Consultancy	poa	APL and Unix system design, consultancy, programming and training.
Evestic AB	Consultancy	poa	Excellent track record from 15+ years of APL applications in banking, insurance, and education services. All dialects, platforms and project phases. SQL expertise.
General Software	Consultancy	from 120	
Godin London Inc	Software Development	poa	We have applications in the food manufacturing field, travel agency and airline bookings field and in product lease management.
Entropy Software Ltd	Consulting	poa	Company reporting, business graphics, Windows applications with Dyalog APL/W.
H.M.W.	Consultancy	poa	System design consultancy, programming. HMW specialize in banking and prototyping work.
Hoekstra Systems Ltd	Consultancy	poa	APL consultancy, programming, etc. Also UNIX system administration
Michael Hughes	Consultancy	poa	Consultant with 10+ years experience with various APL interpreters and C.
IAC/Human Interfaces	Consultancy	poa	APL on Macintosh & PC. HCI design. VDU ergonomics: EC/Health & Safety compliance.
	Documentation	100-200	On-line assistance, product demos & mock-ups, manual writing; foreign language software localization.
	Training	poa	Using I-APL for courseware & distance learning materials; Mac programming in C, APL & HyperCard.

INFOSTROY	Consultancy	poa	Moving applications between platforms. Client/server development. Multilingual user interface.
Insight Systems	Consultancy	poa	Experts in APL conversions between any combination of: APL*PLUS, APL2, Dyalog APL and Sharp APL. We are also experienced right-sizers, comfortable with networks and relational databases (that also means when NOT to use SQL) and client/server development in APL, C and Visual Basic.
JAD Software	Consultancy	poa	Systems design and development, project management, technical manuals, financial and actuarial expertise in APL.
Phil Last	Consultancy	poa	APL consultancy, modelling and programming.
Lescasse Consulting	Consultancy	poa	A range of consultants, experts in Windows programming, with APL+Win and Dyalog APL/W. More than 100 major APL applications already developed. We all have additional expertise in Formula One and Delphi.
Mackay Kinloch Ltd	Consultancy	200-400	Design, analysis and programming for banking, insurance, financial planning and modelling, corporate performance and legal reporting
MicroAPL	Consultancy	poa	Technical & applications consultancy.
Ellis Morgan	Consultancy	250-500	Business Forecasting & APL Systems.
Oasis Systems	Consultancy	poa	Expertise in APL system design, Project management, conversion, migration, tuning; for all APL versions (10+ years experience)
Object Oriented Ltd	Consultancy	poa	General APL consulting, code recycling — mainframe to PC, performance tuning.
Optima	Consultancy	poa	A range of consultants specialising in all areas of pharmaceutical, industrial and financial systems with 5-15 yrs experience on both PC and mainframe.
RadSys Technologies	Consultancy	poa	Areas of expertise: financial systems, risk analysis systems, healthcare systems.
RE Time Tracker Oy	Consultancy	poa	APL application conversions, APL Windows interfaces, APL to API-level interfacing to any system under Windows, TCP/IP network and database connectivity.
Rex Swain	Consultancy	poa	Independent consultant, 20 years experience. Custom software development & training, PC and/or mainframe.
Rochester Group	Consultancy	poa	Specialise in MIS using Sharp APL
Shepp & Associates	Consultancy	poa	APL applications development and consulting, especially in the travel industry, especially on small computers. 25 years experience in APL programming.
Snake Island Research Inc	Consultancy	poa	APL interpreter and compiler enhancements, intrinsic functions, performance consulting. APL parallel compiler APEX is giving very good initial performance tests with convolution somewhat faster than FORTRAN.
Strand Software	Consultancy	poa	Advice on migrating to and from all flavours of APL and hardware platforms. Full-screen interface implementation, APL utilities, benchmarking, efficiency analysis, actuarial software, system development tools, valuation, pricing and modelling systems.
Sykes Systems Inc	Consultancy	poa	Complete APL services specialising in audit, optimisation and conversion of APL systems. Excellent design skills. All dialects and platforms. 17-23 years experience.
Stephen Wynn	Consultancy	poa	Most experience of financial planning, and mathematical areas: operational research, quality control, experimental design.

OTHER PRODUCTS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Adfee	Employment	poa	Contractors and permanent employees
APL-385	Typefaces	poa	Variants of the APL2741 typeface available to specification.

Bloomsbury Software	Training	poa	Contact the company for details.
ComLog	Comic-Logger	$25.95+p&p	APL*PLUS II comic-book inventory system. Shareware version available on America OnLine.
HMW	Employment	poa	Contractors and permanent employees placed.
HRH Systems	APL lessons		On-screen interactive APL lessons for APL*PLUS, TryAPL2, Sharp and I-APL — in English or French.
	The BBS\APL:	$24 p.a.	703-528-7617, 1200-14400b, N-8-1, 24 hours. APL educational material is downloadable free. An additional 30 megs of APL software for APL*PLUS, PLUS II, IBM, Sharp & I-APL is available to subscribers (cost is $24/yr). Selection available on disk for $15 post-paid. Free on-disk catalogue.
I-APL Ltd	An APL Tutorial	3	45pp by Alvord & Thomson
	An Encyclopaedia of APL (2d Ed)	6	228pp by Helzer
	APL in Social Studies	3	36pp by Traberman
	I-APL Instruction Manual (2d Ed)	3	55pp by Camacho & Ziemann
	APL Programs for the Mathematics Classroom (Springer-Verlag)	16	185pp by Thomson
	Programming In J	10	75pp by Ken Iverson
	Arithmetic	12	118pp by Ken Iverson
	Tangible math	8	36pp by Ken Iverson
	Sharp APL Reference Manual	42	349pp by Berry
	APL Press Books	poa	A comprehensive selection of early APL literature
	Please note there is a packing charge of £3 per order		
Oasis Systems	Training	poa	Introductory courses in APL Advanced courses for different APL versions
Renaissance Data Systems	Booksellers		The widest range of APL books available anywhere. See Vector advertisements.
Soliton Associates	MVSLINK	poa	Interface from Sharp APL (Unix & MVS) to non-APL data and software in the MVS environment.
	SSQL	poa	High-performance DB2 interface for Sharp APL (Unix and MVS).

OVERSEAS ASSOCIATIONS

GROUP	LOCATION	JOURNAL	OTHER SERVICES	Ann.Sub.
APL Bay Area	USA N. California	APLBUG	Monthly Meetings (2nd Monday)	$20
APL Club Austria	Austria	-	Quarterly Meetings	200AS(indiv), 1000AS(corp)
APL Club Germany	Germany	APL Journal	Semi-annual meetings	DM60
Ass. Francophone pour la promotion d'APL	France	Les Nouvelles d'APL		FF350 (private) FF2800 (Company)
BACUS	Belgium	APL-CAM	Conferences & Seminars	£18 ($30)
Capital PCUG	Washington, D.C.	Monitor	Monthly meetings, occasional classes	free
Danish SIG	Denmark			
Dutch APL Assoc.	Holland	-	Mini-congress, APL ShareWare Initiative	
FinnAPL	Helsinki, Finland	FinnAPL Newsletter	Seminars on APL	100FIM(private), 30(student), 1000 (Co)
Rome/Italy SIG	Roma, Italy			
RusAPL	Moscow, Russia	APL Club	Seminars and Annual Meeting	100,000R (students 20,000)
SE APL Users Grp	Atlanta, Georgia	SEAPL Newsletter	Quarterly meetings	$10
SovAPL	Obninsk, Russia			
SwedAPL	Sweden	SwedAPL Nytt	Semi-annual meetings, seminars	SEK 75
SWAPL	Texas, USA	SWAPL		$18
Swiss APL (SAUG)	Bern	Part of Qtly SI-Info		SF60 (SI) + SF20 (SAUG)
Toronto SIG	Toronto, Canada	Gimme Arrays!	Monthly Meetings, APL skills database, J SIG, Toronto Toolkit	$25

ADDRESSES

ORGANISATION	CONTACT	ADDRESS, TELEPHONE, FAX, EMAIL etc.
ADAPTA Software GmbH	Michael Baas	Marienhoehe 86, 25451 Quickborn, Germany Tel: +49 4106 60977 Fax: +49 4106 67869 Email: 101523.1757@compuserve.com
Adaptable Systems	Lois & Richard Hill	49 First Street, Black Rock 3193, Australia. Tel: +61 3 9589 5578 Fax: +61 3 9589 3220 Email: adsys@ibm.net
Adaytum Software	Douglas Rowley	13 Great George Street, BRISTOL BS1 5RR UK Tel: 0117-921 5555
Adfee	Bernard Smoor	Dorpsstraat 50, 4128 BZ Lexmond, Netherlands. Tel +31 347 342 337 Fax: +31 347 342 342 Email: adfee@concepts.nl
Ajay Askoolum	Ajay Askoolum	42 Hanworth Road, Redhill, Surrey RH1 5HT Tel:01737 771643 Email: 106173.3347@compuserve.com
Andrews	Dr Anne D Wilson	12 Thorny Hills, Kendal, Cumbria LA9 7AL, UK Tel: 01539-731205
APL-385	Adrian Smith	Brook House, Gilling East, York YO6 4JJ UK. Tel: 01439-788385 Fax: 01439-788194 Email: 100331.644@compuserve.com
APL Bay Area APLBUG	Curtis Jones (Sec)	228 South 15th Street, San Jose, CA 95112-2150, USA Tel: +1 (408) 292-4060 Email: jonesca@vnet.ibm.com
APL Club Austria	Harald F. Nelson	c/o N-TECH, Siebenbrunnenfeldg. 4-6, A-1050 Wien, Austria. Tel: +43 1 5458063 Fax: +43 1 5458063-17
APL Club Germany	Dieter Lattermann	Rheinstraße 23, D-69190 Walldorf, Germany. Tel: +49 6227-63469 Compuserve: 100332,1461
The APL Group Inc	Stuart Sawabini	644 Danbury Road, WILTON, CT 06897 USA. Tel: +1 (203) 762-3933 Fax: +1 (203) 762-2108 Email: ssawabini@aol.com, eshaw@aplgroup.com
APL Solutions Inc	Eric Landau	1107 Dale Drive, Silver Spring, MD 20910-1607 USA Tel: +1 (301) 589-4621 Fax: +1 (301) 589-4618 Email: elandau@cais.com
Association Francophone pour la promotion d'APL	Ludmila Lemagnen	174 Boulevard de Charonne, F-75020 Paris, FRANCE Email: lemagnen@aol.com
AUSCAN Software Ltd	Richard Procter	8 Springmount Ave, Toronto, Ontario M6H 2Y4 Canada Tel: +1-416-651-4037 Email: rjp@interlog.com Web: http://www.interlog.com/~rjp/auscan/
BACUS	Joseph De Kerf	Roolnberg 72, B-2570 Duffel, Belgium. Tel: +32 15 31 47 24
Beautiful Systems, Inc.	Jim Goff	308 Old York Road, Suite 5, Jenkintown, PA 19046, USA Tel: +1 (215) 886-2636; Fax: +1 (215) 886-4888
Bloomsbury Software	Peter Day	3-6 Alfred Place, Bloomsbury, London WC1E 7EB UK. Tel: 0171-436 9481 Fax: 0171-436 0524 Email: pd@bloomsbury-software.co.uk
Camacho	Anthony Camacho	11 Auburn Road, Redland, Bristol BS6 6LS UK. Tel: 0117-9730036. email: acamacho@cix.compulink.co.uk Reuternet (Sharp): ACAM
Ray Cannon		21 Woodbridge Rd, Blackwater, Camberley, Surrey GU17 0BS UK Tel: 01252-874697 Email: 100430.740@compuserve.com
Paul Chapman		51B Lambs Conduit Street, London WC1N 3NB UK. Tel: 0171-404 5401. Compuserve: 100343,3210
Causeway Graphical Systems Ltd	Adrian Smith	The Maltings, Castlegate, MALTON, North Yorks YO17 0DP UK Tel: 01653-696760 Fax: 01653-697719 Email: causeway@compuserve.com
Cinerea AB	Rolf Kornemark	Box 61, S-193 00 Sigtuna, Sweden. Tel/Fax: +46 859 255 421 Email rolf@cinerea.se
CODEWORK Italia srl	Mauro Guazzo	Corso Cairoli 32, 10123 Torino, Italy. Tel: +39 11 885168 Fax: +39 11 812 2652 Email: codework@inrete.it
ComLog Software	Jeff Pedneau	PO Box 5570, Derwood, MD 20855 USA Tel: +1 (301) 990-7063 Email: jeff@softmed.com
CPCUG	Lynne Sturtz	Capital PC User Group, 51 Monroe Street, Suite PE-2, Rockville, Maryland 20850-2421, USA. Tel: +1 (301) 762-9372 Fax: (301) 762-9375.
Danish User Group	Per Gjerløv	Email: gjerlov@ibm.net
Dinosoft Oy	Pertti Kalliojärvi	Lönnrotinkatu 21C, 00120 Helsinki, FINLAND. Tel: +358 9 70028820 Fax: +358 9 70028824 Email: dinosoft@dinosoft.fi
Dutch APL Association	Bernard Smoor (Sec)	Postbus 1341, 3430BH Nieuwegein, Netherlands. Tel: +31 347 342 337 Fax: +31 347 342 342
Dyadic Systems Ltd.	Peter Donnelly	Riverside View, Basing Road, Old Basing, Basingstoke, Hants RG24 0AL UK. Tel: 01256-811125 Fax: 01256-811130

Entropy Software Ltd	George MacLeod	Bartrum House, Ravens Lane, Berkhamsted, Herts, HP24 2DY UK Tel: 01442-878065 Email: gml@simcorp.co.uk
Evestic AB	Olle Evero	Bertellusvagen 12A, S-146 38 Tullinge, Sweden Tel&Fax: +46 778 4410 Email: olle.evero@mailbox.swipnet.se
FinnAPL		Suomen APL-Yhdistys RY, FinnAPL RF, PL 1005, 00101 Helsinki 10, Finland
General Software Ltd	M.E. Martin	22 Russell Road, Northholt, Middx, UB5 4QS UK. Tel/fax: 0181-864 9537
Godin London Incorporated	Gaëtan Godin	12 Gerrard St., London, Ontario, Canada N6C 4C5 Tel: +1 (519) 679-8290 Fax: +1 (519) 438-6381 Email: info@godin.on.ca
H.M.W.Trading Systems Ltd		Hamilton House, 1 Temple Avenue, Victoria Embankment, London EC4Y 0HA UK. Tel: 0171-353 8900; Fax: 0171-353 3325; Email:100020.2632@ compuserve.com
Hoekstra Systems Ltd	Bob Hoekstra	5 Thorsden Court, Guildford Road, Woking, Surrey, GU22 7QS UK Tel: 01483-771028 Email: bob@khamsin.demon.co.uk
HRH Systems	Dick Holt	3802 N Richmond St, Suite 271, Arlington, VA 22207 USA Tel: +1 (703) 528-7624; Email: dick.holt@acm.org
Michael Hughes		28 Rushton Road, Wilbarston, Market Harborough, Leics. LE16 8QL UK. Tel: 01536-770998 Email: 101740.1203@compuserve.com
IAC/Human Interfaces	Ian A. Clark	9 Hill End, Frosterley, Bishop Auckland, Co. Durham DL13 2SX UK Tel: 01388-526803, Compuserve: 100021,3073
I-APL Ltd	Anthony Camacho (for queries, order forms)	11 Auburn Road, Redland, Bristol BS6 6LS UK. Tel: 0117-9760036 email: acamacho@clx.compulink.co.uk Reuternet (Sharp): ACAM
	J C Business Services (for pre-paid orders only)	56 The Crescent, Milton, Weston-super-Mare, Avon, BS22 8DU UK Tel: 01934-625181
IBM APL Products	Nancy Wheeler	APL Products, IBM Santa Teresa, Dept M46/D12, 555 Bailey Avenue, San Jose CA 95141, USA. Tel: +1 (408) 463-APL2 (=2752) Fax: +1 (408) 463-4488 Email: APL2@vnet.ibm.com Cserve: GO IBMAPL2
INFOSTROY	Alexei Miroshnikov	3 S. Tulenin Lane, St. Petersburg 191186 Russia. Tel:+7 812 312-2673 Fax:+7 812 311-2184 Email:alm@infostroy.spb.su
Insight Systems ApS	Morten Kromberg	Nordre Strandvej 119C, DK-3150 Hellebæk, Denmark Tel:+45 49 76 20 20 Fax: +45 49 76 20 30 Email: info@insight.dk
Iverson Software Inc.	Eric Iverson	33 Major Street, Toronto, Ontario, Canada M5S 2K9 Tel: +1 (416) 925-6096; Fax: +1 (416) 488-7559 Email: info@jsoftware.com
J Austria	Joachim Hoffmann	Münzgrabenstr. 68, A-8010 Graz, Austria. Tel: +43 (0)316 814529 Fax: +43 (0)316 816683 Email: JoHo@ping.at
JAD Software	David Crossley	580 Eyer Drive, #81 Pickering, Ontario, Canada L1W 3B7 Tel: +1 (905) 837-1895 Fax: +1 (905) 831-5172
Phil Last Ltd	Phil Last	146 Crossbrook Street, Cheshunt, Herts, EN8 8JY UK. Tel: 01992 633807 Fax: 0121 359 0375 Email: phil_last@compuserve.com
Lescasse Consulting	Eric Lescasse	18 rue de la Belle Feuille, 92100 Boulogne, France Tel: +33.1.46.05.10.76 Fax: +33.1.46.04.60.23 Email: eric@lescasse.com
Lingo Allegro USA, Inc	Steven J Halasz	1105 Chicago Avenue, Suite 155, Oak Park, IL 60302, USA. Tel:+1 708 386 8183 Email: sjhalasz@interaccess.com
Mackay Kinloch Ltd	Alastair Kinloch	519 Webster's Land, Edinburgh EH1 2RX, Scotland, UK Tel/Fax/Answerphone: 0131 228 3580 Pager/Voicemail: 01426 98 3858 Email: Alastair_Kinloch@compuserve.com
Mercia Software Ltd.	Gareth Brentnall	Holt Court North, Heneage Street West, Aston Science Park, Birmingham B7 4AX UK. Tel: 0121-359 5096. Fax: 0121-359 0375
MicroAPL Ltd.	Richard Nabavi	South Bank Technopark, 90 London Road, LONDON SE1 6LN UK Tel: 0171-922 8866 Fax: 0171-928 1006 Email: MicroAPL@microapl.demon.co.uk
Ellis Morgan	Ellis Morgan	Myrtle Farm, Winchester Road, Stroud, Petersfield, Hants GU32 3PE UK. Tel: 01730-263843 Email: Ellis@mrtlfrm.demon.co.uk
Oasis Systems B.V.	Theo Zwart, Louis Rijkse	Lekstraat 4, 3433 ZB Nieuwegein, Holland Tel: +31 30 60 66 336 Fax: +31 30 60 65 844 Email: oasisbv@pi.net or zwart@oasis.nl
Object Oriented Ltd	Walter G. Fil	Am Grendel 2, CH-6004 Luzern, Switzerland. Tel: 41 41 418 70 70 Fax: 41 41 418 70 77 Email: info@object-oriented.com
Optima Systems Ltd	Paul Grosvenor	115 Brighton Road, Purley, Surrey CR8 4HE UK Tel: 0181-763 2490 Fax: 0181-763 2491 Email: 100551.1401@compuserve.com

RadSys Technologies AB	Randolph Schrab	Lovsangarv. 18, S-756 52 Uppsala, Sweden. Tel: +46 18 32 41 53 Fax: +46 708 1996 11 Email:100564.2544@compuserve.com
Renaissance Data Systems	Ed Shaw	PO Box 421, Georgetown, CT 06982, USA. Tel: +1 (203) 270-9729
RE Time Tracker Oy	Richard Eller	Mikonkatu 8 A, 2.krs, PL 363, 00101 Helsinki, Finland. Tel: +358 9-621 3300 Fax: +358 9-621 3378 Email: re@rett.fi
The Rochester Group Inc.	Robert Pullman	50 S.Union St., Rochester NY 14607-1828, USA. Tel: not known. Fax: +1 (716) 271-1230
Rome/Italy SIG	Mario Sacco	Casella Postale 14343, 00100-Roma Trullo, Italy Email: marsac@vnet.ibm.com
RusAPL	Boris Makeev	box 971 (for Makeev), Dmitrovskoe Sh.,2, 127434, Moscow, Russia Tel/fax: +7 95 210-7783 Email: makeev@atom.ai.x-atom.net
SE APL Users Group	John Manges	413 Comanche Trail, Lawrenceville, GA 30044, USA Tel: +1 (770) 972-3755 Email: seapldoc@aol.com
Shepp & Associates LLC	Andrew Shepp	1312 Washington Avenue, 6th Floor St. Louis MO 63103, USA Tel: +1 (314) 621-3272 Fax: +1 (314) 621-4267 UK Address: Claridge House, 29 Barnes High St, London SW13 9LW Tel: 0181 8768666 Fax: 0181 8768660
Snake Island Research Inc	Bob Bernecky	18 Fifth Street, Ward's Island, Toronto, Ontario M5J 2B9 Canada Tel: +1 (416) 203-0854 Fax: +1 (416) 203-6999 Email: bernecky@eecg.toronto.edu
SOCAL (South California)	Roy Sykes Jr	Sykes Systems Inc, 4649 Willens Ave, Woodland Hills, CA 91364-3812 USA Tel: +1 (818) 222-2759 Fax: +1 (818) 222-9250
Soliton Associates	Laurie Howard	Soliton Associates Ltd, Groot Blankenberg 53, 1082 AC Amsterdam, Netherlands Tel: +31 20 646 4475 Fax: +31 20 644 1206 Email:sales@soliton.com
SovAPL	Alexander Skomorokhov	PO Box 5061, Obninsk-5, Kaluga Region, Russia Tel: +7(08439)31463 Email:askom2@kaluga.rosmail.com
Strand Software Inc	Anne Faust	19235 Covington Court, Shorewood MN 55331 USA Tel: +1 (612) 470-7345 Email: amfaust@aol.com
Rex Swain	Rex Swain	8 South Street, Washington, CT 06793 USA. Tel: +1 (860) 868-0131 Fax: +1 (860) 868-9970 Email: rhswain@acm.org
SwedAPL	Christer Ulfhielm	Novator Consulting Group AB, Svärdvägen 11C, S-182 33 Danderyd Sweden Tel: +46 8 622 63 50 Fax: +46 8 622 63 51 CServe: 100341,404
Swiss APL User Group		Swiss APL User Group, CH-3001, Bern 1, Switzerland Email: si@ifi.unizh.ch
Sykes Systems Inc	Roy Sykes Jr	4649 Willens Ave., Woodland Hills, CA 91364, USA Tel: +1 (818) 222-2759 Fax: +1 (818) 222-9250
Toronto SIG	Richard Procter	PO Box 55, Adelaide St. Post Office, Toronto Ontario M5C 2H8, Canada Email: rjp@interlog.com www.sigapl.mtnlake.com/sigapl/welcome.html
Stephen Wynn		8 Clarence Gardens, Brighton, Sussex BN1 2EG Tel: 01273-327238 Email: centre@mistral.co.uk
Zark Incorporated	Gary A. Bergquist	23 Ketchbrook Lane, Ellington CT 06029, USA. Tel: +1 (860) 872-7806

WORLD WIDE WEB SITES

ORGANISATION	URL
AFAPL	www.ensmp.fr/~scherer/langlet/ (Journal available on line)
APL2000	www.APL2000.com
APL-385	www.demon.co.uk/apl385
The APL Group Inc	www.aplgroup.com
AUSCAN	www.interlog.com/~rjp/auscan
Capital PC User Group	http://cpcug.org
Causeway	www.causeway.co.uk
CODEWORK	www.codework.de
COSY (Bob Armstrong)	www.cosy.com
Dinosoft Oy	http://yritys.kolumbus.fi/dinosoft
Dyadic Systems Ltd	www.dyadic.com
FinnAPL	http://personal.eunet.fi/pp/apl
Godin London Inc	www.godin.com
IBM APL2	www.torolab.ibm.com/ap/apl/apl2.html
Infostroy	www.insight.dk/infostroy
Insight Systems ApS	www.insight.dk
Iverson Software Inc	www.jsoftware.com
Kestrel Consulting	www.kestrel-consulting.com
Lescasse Consulting	www.lescasse.com
Lingo Allegro USA Inc	www.lingo.com
Mackay Kinloch Ltd	ourworld.compuserve.com/homepages/Alastair_Kinloch
MicroAPL Ltd	www.microapl.co.uk
RE Time Tracker Oy	www.rett.fi
Shepp & Associates	www.digitravel.com
SigAPL	www.acm.org/sigapl
Rex Swain	www.pcnet.com/~rhswain
The APL Group Inc	www.aplgroup.com
Toronto SIG	www.sigapl.mtnlake.com/sigapl/welcome.html
Jim Weigang	www.chilton.com/~jimw

FTP SITES

ORGANISATION	DOMAIN NAME
IBM APL2	ps.boulder.ibm.com/ps/products/apl2/
Toronto toolkit	see Toronto SIG home page
Waterloo Archive	archive.uwaterloo.ca/ftparch/languages/apl
APL-to-ASCII	archive.uwaterloo.ca/languages/apl/workspaces/aplascii

THE EDUCATION VECTOR

The Josephus Problem Generalized in J

by Howard A. Peelle (hapeelle@educ.umass.edu)

Programs for the generalized Josephus problem and variants are presented using J.

Introduction

The Josephus problem is a classic mathematical problem which has been portrayed in various guises and has been programmed in various languages (see [1]), typically for an assignment in computer science. It supposedly originated in the first century when 41 trapped rebels stood in a circle and, rather than be captured, sacrificed every third person remaining. Flavius Josephus and a friend apparently were the two survivors [2]. The question is: where did they stand (indexed from the start position)?

The problem invites generalization in two variables: *N*, the number of people to start with; and *K*, the interval for killing. Simulations can be programmed easily; for instance, from [3]:

```
    j =: }: @ |.

    js =: j ^: (1: + #@] - [)
```

For example, the original case above:

```
    3 js i.41
  15 30                          NB. Survivors
```

`K js i.N` works for any integer *N* when *K*=2 but not for *N*<*K*-1 when *K*>2; therefore, `K&js\ i.N` yields a pattern of results only for *K*=2. Degenerate case `1 js i.N` is empty (correctly) and `0 js i.N` is empty (but should be `i.N`). When

K<0, the Josephus problem can be interpreted as killing in the opposite direction, which can be mapped into positive *K* cases: `(|K) js (-K<0)|.i.N**K .`

Accordingly, all programs here are defined for *K*>0 and with *N*>0 in mind, although most will work for any integer *N*, where:

```
    K js i.-N is N - >:K js i.N .
```

The following simulation program accommodates any positive integers *K* and *N* (including degenerate cases), and uses input indices for convenience:

```
    JS =: j ^: (<: #) ^:_ i.

    3 JS 41                        NB. Kill every 3rd of 41
 15 30                             NB. Survivors (origin 0)
```

`K JS >:i.N` simulations produce results generally useful for analyzing patterns.

Analytic solutions, however, are rare. A nice closed form solution for *K*=2 exists: represent *N* in binary, rotate the first digit onto the end, and decode the binary representation. The following function (from [4] and derived in [2]) accomplishes this:

```
    Josephus =: 1&|. &. #:
```

For example:

```
    Josephus 41                    NB. Kill every 2nd of 41
 19                                NB. Survivor (Origin 1)
```

An alternative is:

```
    Josephus =: (- -.) &. #:
```

Unfortunately, neither lends an obvious generalization for *K*>2. Instead, recurrence relations have been proffered for *K*=3 and for *K*>1 in [2 pages 80-81], as well as for *K*=2 in [5 page 76] and [3 page 11]. Other non-recursive solutions, such as in [3 page 12] for *K*=2, apparently haven't been generalized.

The General Josephus Problem

The generalized Josephus problem is restated here as follows:

> For *N* people in a circle, if every *K*th remaining person is successively killed, what were the original positions of the *K*-1 survivors?

Here is a simple recursive program to solve this problem:

```
    k =: [
    n =: ]
    J =: (n | k + (J <:)) ` (i.@n) @. >
```

Examples:

```
    2 J 41
 18                              NB. Origin 0

    3 J 41
 15 30

    4 J 41
 36 10 4
```

Results of `K J"0 >:i.N` replicate (albeit inefficiently) `K JS"0 >:i.N` simulations.

General Josephus Problem Variants

A common variant of the General Josephus problem continues the killing until no one remains and reports all killed in order, as in [6]. The following simulation program does this recursively:

```
 JO =: ({:@|. , k JO }:@|.) ` n @. (#@n <: 1:)
```

For example:

```
    3 JO i.41                    NB. Order of killing
 2 5 8 11 14 17 20 23 26 29 32 35 38 0 4 9 13 18 22 27 31
    36 40 6 12 19 25 33 39 7 16 28 37 10 24 1 21 3 34 15 30
```

Or, to report when each person was killed, use the inverse permutation:

```
    /: 3 JO i.41                 NB. When each killed
 13 35 0 37 14 1 23 29 2 15 33 3 24 16 4 39 30 5 17 25 6 36
    18 7 34 26 8 19 31 9 40 20 10 27 38 11 21 32 12 28 22
```

A simpler variant of the General Josephus problem reports only the last survivor, as in [2]. This can be accomplished easily using the previous program. For example:

```
    {: 3 JO i.41
 30
```

Other Variants

Now consider some other variants of the General Josephus problem. First:

> For *N* people in a circle, if the first of every *K* remaining people is successively killed, what were the original positions of the *K*−1 survivors?

A simulation program for this problem is:

```
    j1 =: }. , }.@{.

    JS1 =: j1 ^: (<: #) ^:_ i.
```

Examples:

```
    2 JS1 41
17

    3 JS1 41
13 28

    4 JS1 41
33 7 11
```

Comparing results of `K JS1"0 >:i.N` with `K J"0 >:i.N` reveals that `JS1` is `>:@(J <:)` for *N*>*K*-1 or `i.@(<: <:)~ , >:@(J <:)` in general.

Next, a complementary variant (for *K*>1):

> For *N* people in a circle, if all but the first of every *K* remaining people are successively killed, what was the original position of the last survivor?

A simulation program for this is `JS_1 =: j_1 ^: (#@] > 1:) ^:_ i.` where `j_1 =: }.,{.@]`.

Better yet, here is a closed form solution to this problem:

```
    k =: [
    n =: ]
    J_1 =: mod * (k (* - power) n%mod) % <:@k

    power =: k ^ >:@<.@^.

    mod =: >:@(|&<:)
```

Examples:

```
   2 J_1 41
18

   3 J_1 41
21

   4 J_1 41
12
```

Finally, another related variant:

> For *N* people in a circle, if all but the *K*th remaining person are successively killed, what was the original position of the last survivor?

Readers are welcome to help solve this problem and to prove the correctness of the programs given here.

References

[1] *Problem Solving by Programming: Solutions to The Josephus Problem (Revisited),* Peelle & Lipp, Journal of Computer Science Education, Vol. 12, No. 1, Fall 1997

[2] *Concrete Mathematics (2nd edition),* Graham, Knuth, & Patashnik, Addison-Wesley, Reading, MA 1994, pages 8–16, 79–81

[3] *Concrete Math Companion,* Kenneth Iverson, Iverson Software Inc., Toronto 1996, pages 10–12

[4] *At Work and Play in the Fields of J,* Eugene McDonnell, Iverson Software, Inc., Toronto 1993, page 16

[5] *Trains, Agendas, and Gerunds,* Hui & Iverson, Proceedings of APL94, Vol. 25, No. 1, ACM, New York, pages 74–77

[6] *Implementing Discrete Mathematics,* Steven Skiena, Addison-Wesley, Redwood City, CA 1990

J-ottings 14

by Norman Thomson

THINK, REASON, EXPRESS was Bill McLean's plea as he struggled to understand the verb `malt`, standing for "merge alternate", which appeared in J-ottings 13. Here is how it was given there:

```
  malt=./:@,&(i.@#){, NB. merge x. and y. using alternate elements
```

In Bill's view, J strings with thirteen characters in the definition do not lend themselves to easy reading, a view with which I would concur! `malt` thus breaks what for me is an implicit rule that a tacit function definition which exceeds about seven to eight characters should be broken down into subsidiary verbs. At least this is a rule which should be observed if there is any expectation that the verb definition should be readily intelligible to a reader with a sound knowledge of the primitive J verbs. There is a close parallel with words in English. A new or neologised seven or eight letter word ought to be intelligible to, or at least pronouncable on sight by, a native speaker, whereas twelve or thirteen letters usually force such a reader to hesitate in piecing together the word's basic construction. I excuse myself concerning `malt` because it was used as an auxiliary verb in solving a larger problem. However it furnishes a good illustration of the principle of "think, reason, express", and in my view thereby deserves further discussion.

First the problem itself is that of merging two vectors (not necessarily of equal length) with items taken alternately from each so that, for example, the merge of `1 2 3 4` and `5 6` is `1 5 2 6 3 4`. "Think" and "reason" might appear to be synonyms — I use the former to describe understanding a problem in terms unrelated to the intended programming language, whereas the latter is about matching the details of the problem to, in this case, the constructs of J. A suitable set of synonyms might be "analyse, model, communicate".

So let's think! It would be reasonable to glue the vectors together as `1 2 3 4 5 6` and then seek an algorithm for the ordered indices which define the merge. (No mention of J as yet!) Now let's reason. "Glue" is modelled by "join" and indexing by "from". The latter is clearly the central operation, following two preliminary data transformations, one (on the right) a join, and the other (on the left) the algorithm which produces the correct indexes. This generic form is what I think Donald McIntyre refers to as "the ubiquitous fork". Since two of the present operations are primtives in J, encouragement can be taken from the fact that two-thirds of the work is already done:

```
malt=.mselect { ,
```

This leaves only the subordinate verb `mselect` to be designed. Once again, think. The result of gluing two index vectors of the same lengths as the data vectors is a vector containing somewhere within it two 0s which match the first pair of items in the merged vector, two 1s matching the next pair, and so on. Thus far no J. Now reason — gluing takes place following a preliminary transformation on each vector, the classic circumstance in which to use "with" (`&`). The transformation from vector to index vector is "tally then index", so define

```
ivector=.i.@#
```

and the gluing is then

```
ijoin=.,& ivector
```

The final step exploits the way in which grade-up works with ties, namely by counting them from left to right occurrences, which leads to the index selection verb

```
mselect=./:@ijoin
```

So combining everything, the analysis and subsequent modelling is expressed by

```
malt=.mselect { ,
mselect=./:@ijoin
ijoin=.,& ivector
ivector=.i.@#
```

With hindsight, J has proved an excellent medium in which to embody the principle of THINK, REASON, EXPRESS.

Of course, when `malt` becomes part of a larger algorithm it is not unreasonable to reduce the number of subordinate verbs in the manner given in J-ottings 13 and the start of this article. With this in mind, it is instructive to consider the `merge` algorithm given in J Phrases, viz:

```
merge=.1 :'/: @ /:@(x."_){,'
```

First this is an adverb, and secondly the definition is explicit. Nevertheless the final two-thirds are identical to `malt`, namely `{,`. As for the rest, the purpose is to generalise matters to allow the merge to be determined by a left argument such as `0 0 1 1 0 0` which would lead to a solution `1 2 5 6 3 4` for the example vectors above.

Next, consider the verbs `over` and `by` which are given as "black boxes" at an early stage of the Introduction and Dictionary:

```
  over=.({.;}.)@":@,
  by=.' '&;@,.@[,.]
  'abc' by 1 2 3 4 over i.3 4

  +----------------+
  |    | 1  2  3  4 |
  |----------------+
  | a  | 0  1  2  3 |
  | b  | 4  5  6  7 |
  | c  | 8  9 10 11 |
  +----------------+
```

both of which fall well outside my seven/eight character comprehensibility limit, and could be accused of giving at least an initial impression of what Don Mattern calls "comic book invective". It is instructive to reverse engineer these verbs into a "think, reason, express" sequence.

The objective of `over` is to attach numeric column headings to a matrix, and so ultimately each headed column has to be presented, suitably formatted, in the form of two vertically aligned boxes. Thus far, no J. Now reason. Join and format are primitives which should be executed in that order, hence

```
      join_fmtd=.":@,
      1 2 3 4 join_fmtd i.3 4
  1 2  3  4
  0 1  2  3
  4 5  6  7
  8 9 10 11
      $1 2 3 4 join_fmtd i.3 4
  4 9
```

In J, when one or more of a pair of joined objects has rank more than 1, the objects are displayed in a vertical stack. Now exploit the knowledge that "link" (`;`) automatically boxes objects. In the present instance the column headers and table body should be boxed separately following the join, that is, two boxes are required, one for the head, and the other for the tail. This leads to another instance of a fork:

```
  boxes=.{.;}.
```

```
   boxes 1 2 3 4 join_fmtd i.3 4

   +--------------------+
   |1 2 3 4 | 0  1  2  3 |
   |        | 4  5  6  7 |
   |        | 8  9 10 11 |
   +--------------------+
```

This operation must follow `join_fmtd` in sequence, leading to

```
   over=.boxes @ join_fmtd
   boxes=.{.;}.
   join_fmtd=.":@,
```

The fact that the boxes are side by side rather than vertically stacked does not matter since the verb `by` will complete the alignment process.

The main purpose of `by` is to add the row headings which may be either character or numeric. This stage must deal with the extra complexity of the "top-left-corner cell" which characterises data presented with row and column headers. On the assumption that this cell is occupied with empty space, it should be boxed and appended to a second box containing the row headings, and the result of this appended columnwise to the result of `over`. Once again, this is all thinking/analysis, no J yet. Next the reasoning stage. The picture at the end of the thinking stage is two boxed two-item columns, the leftmost containing space and row headers, and the rightmost column headers and the table body. The correct display is an itemwise append of these objects which is given by append items `,.` . The underlying generic form is the same as in `malt`, namely a fork whose central operation is "append items", and in which two-thirds of the work is done by primitives, leaving the enlargement of the row header outstanding as a subproblem:

```
   by=.enlarge_rowhdr ,. ]
```

The enlargement of the row header with space is also modelled by append items `,.` , (a simple join would result in an object whose tally is `r+1` where `r` is the number of rows), and, as in `over`, the definition of link guarantees the required boxing. Itemwise appending should be done "with" link, that is using "with" (`&`) in its technical J sense:

```
   space_append=.' '&;@,.
   space_append 'abc'

   +-------+
   |  | a |
   |  | b |
   |  | c |
   +-------+
```

When space_append is integrated into the verb by, the row header becomes the left argument, and so enlarge_rowhdr is simply space_append sitting atop [with the right argument not involved:

```
   'abc'(space_append@[) i.3 4

   +-------+
   |  | a |
   |  | b |
   |  | c |
   +-------+
```

The expression of the reasoned definition is thus:

```
   by=.space_append@[ ,. ]
   space_append=.' '&;@,.
```

In the "black-box" definition of by, the implied bracketing is

```
   over=.(({.;}.)@":)@,
   by=.(' '&;@,.)@[),.]
```

on account of the rule that conjunctions have long left scope, that is, in the absence of parentheses, the occurrence of the first verb to the right of say @ forces a composition. In over as developed above, boxes sits atop the composition of join and format, whereas in the "black-box" version, the composition of boxes and format is atop join. Since these particular verbs obey an associative law, that is (f@g)@h is equivalent to f@(g@h), the two versions of over are functionally identical.

APL97 AT TORONTO

Conference Notes, Photos and Selected Papers

Acknowledgements

The conference notes are mainly by Duncan Pearson, with some extra material from Adrian Smith. The photographs are selected from the very comprehensive set by Richard Proctor.

The three papers which follow have been revised by their authors subsequent to the conference, but are not materially different from the material which appears on the APL98 CD ROM.

You may order the APL97 CD ROM from www.torontoapl.org for $C20, postage paid world-wide, via cheque, money order, Visa, Master, or Amex. Toronto SIGAPL will make the CD ROM electronically available and updated at, or linked to, its web page (www.torontoapl.org).

Vector 14.3 will carry a full directory listing of the CD, with abstracts of papers where available.

APL97 AT TORONTO

Conference Report

by Duncan Pearson and Adrian Smith

Opening Address (Ian Sharp)

This talk was for many the highlight of the conference, probably most for the wit, poise and excellent comic timing of Ian Sharp. Using the parallel of the pocket knife he attacked the tendency of the writers of interpreters to add "useful" features to what was a simple elegant tool and made a proposition that each major new feature, each extra blade on the knife, halved its market, even though the new "improved" version was fully "upwardly compatible" with the old.

Ian Sharp with an OBPK

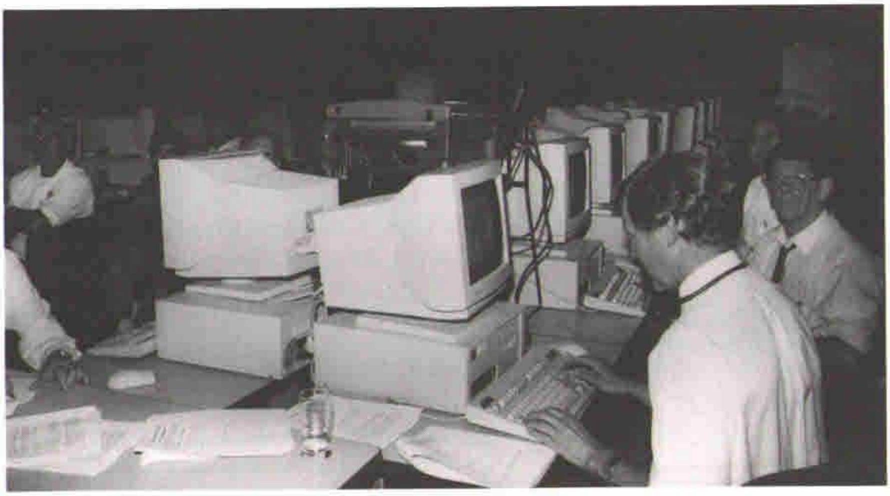
He may have had a point and his speech was very well received, perhaps by many who had been programming in 2nd generation APL too long to remember quite how painful some things could be without the benefit of nested (or boxed) arrays.

"Remember – every time you add a new feature to your product, you reduce its potential market by half."

Perhaps the less well remembered part of his talk was about the role of the refrigerator with its post-it notes, rather than the television, as the family information centre. He pointed out that a great deal of what we see as advanced computing and user-interface techniques have been going on in people's kitchens for many years already, such as overlapping windows and garbage collection (on notes) using a complicated combination of least-recently-used and time-expired algorithms.

Dyalog APL Socket Programming

This was a workshop using the one laboratory of networked PCs. Peter Donnelly introduced sockets and TCP/IP in the context of the Dyalog APL implementation which is well integrated into their GUI object/message structure with socket objects of different types and events occurring on them, to which callback functions can be attached. Sockets on different ports, which might be implementing different interfaces to the same data, would have the functions specific to each interface as callbacks on the events of the sockets for that interface. The event functions might themselves reside in the namespace of the socket. This contrasts with the J approach which provides a very thin cover on the socket API through a foreign conjunction, and a number of cover functions to simplify most of the standard tasks. The J approach might provide a finer control but it would be easier for the tyro to get going with the Dyalog implementation as we saw from the success of the exercises.


Peter Donnelly giving his Workshop

Everyone ran through a series of exercises that developed through two interpreters on the same machine passing text messages to each other, to interpreters on separate machines exchanging APL arrays. The pace of the exercises was well judged with very few users falling behind and with plenty for the more adventurous to get their teeth into. There were few of the technical glitches that have dogged this kind of event in the past and the whole thing seemed to run very smoothly.

Bob Bernecky – APEX Compiled APL

In this talk Bob went through the reasons for compiling APL and then described some of the techniques that he uses to provide improvements in execution time.

The main reason for the whole exercise being speed Bob went into just what takes the time in normal APL execution. From runtime analysis of the Sharp APL timeshare system and from other studies he gave statistics that only a quarter of all time taken was spent in the primitive functions and that the rest was distributed between syntax analysis, argument conformance checking and memory allocation. The one surprise to those who have not attended one of Bob's talks is that in nearly all APL systems, whether they are well or badly written, the majority of code is working on one- or two-element arrays and that the standard argument for APL, that the interpreter overhead is compensated for by the small amount of interpretation relative to work done, is simply not valid in reality.

The second and to me more interesting part of the talk concentrated on some of the techniques used to reduce the time taken in not only the syntax analysis but also in the conformance checking and memory allocation, and even in some cases by the use of clever optimisations in the execution. The APEX compiler works on a self-contained unit of code, usually a function and its sub-functions, for which the inputs and outputs are known. It analyses code in files in the APLASCII Workspace Interchange Format (`.vif`). It determines for each noun in the system as much information about it as it can, for each stage in the process. If on one line a variable is assigned a permutation of an index vector using deal on equal arguments, then on subsequent lines where that variable appears the rank and datatype are known, provided that there are no branches of execution in between the assignment and the use. If the assignment used a constant argument to the deal then not only is the rank but also the shape known. This allows pre-allocation of memory. If the fact that it is a permutation vector is also carried forward and a subsequent use of the vector is to sort it, then an optimised single-pass sort that uses a pigeon-hole technique can be used providing further reductions in execution time.

These methods are obviously heavily dependent on being certain of the possible execution paths through the code and much of the work that Bob has done is in this area where he uses state-of-the-art analysis to determine this as reliably as possible.

The output of the analysis of the APL is not machine code, nor even C, but SISAL which is a language optimised for parallel execution on multi-processor machines. This in its turn generates C which is compiled into the object code. The opening plenary at APL95 was given by the leader of the SISAL project at the Lawrence Livermore Laboratories. For those compiling APL, speed is obviously important and to such people the best use of multi-processor machines is likely to be a significant requirement.

Ken Iverson and Roger Hui – The Mathematical Roots of J

This talk, particularly the second half, was for me the highlight of the conference. Ken and Roger did a double act with Ken on the foils doing most of the talking and with Roger on the keyboard interjecting with more detail and providing interesting examples using the new J Version 3.05.

Ken and Roger

I had expected an abstruse talk on the domain theory underlying the J syntax but in fact the talk was more about the way in which APL and J have their conceptual roots in mathematical ideas and notation. The talk had the title it did because nearly all of the examples, especially the more detailed ones, were from J. One explanation that stuck in my head illustrates these roots well. The constant verb, which to a programmer seems a superfluous thing, is seen in context when considering for example the third derivative of a cubic which, despite being constant, is still a function.

The talk was wide ranging and covered the main explanatory areas of the dictionary pretty much in order. There were explanations of the intrinsic ambivalence of all J verbs, the cap, the fork (covering also the dyadic case which was missed in Donald MacIntyre's talk at APL96) and also the way in which verbs act on arrays using rank, including an explanation of cells frames and atoms. There was a diversion into history to discuss the etymology of matrix and vector and then a discussion of roots of unity at which point Roger said that he had some examples. He dazzled us with some neat expressions demonstrating the isomorphism of the group of seventh roots under multiplication to iota 7 under + mod 7. Ken then went on to give an explanation of Dual and its similarity to matrix operations involving transpose and inverse.

After some more examples from Roger we moved into the seriously mathematical realm of Stirling numbers, coefficients of the Taylor expansion — which I thought was extremely well explained, and Fibonacci numbers. About here I got a little lost but I retained a feeling that my time had been well spent and that J was supremely well suited to supporting the teaching of mathematics in high schools. It is good that this market is now being explored by Strand and ISI.

John Scholes – Dynamic Functions

The use of dynamic functions has been explained in Vector already after they made their appearance at APL96 so I will not go into the details here. John gave us some of the background, explaining how he had caught the functional programming bug and had dreamed of incorporating the techniques into APL. He gave some simple examples of the syntax and demonstrated recursion and also the use of tail recursion to simulate looping. He has put the same optimisation into Dyalog APL as appears in Scheme and many other LISP dialects. If the last line of a dynamic function is a call to itself, after which no more work needs to be done in the function, then the function need not create a new level of the stack, but merely take its arguments back up to the beginning again. This and the lexical scoping, whereby free variables are bound at definition time rather than at execution time, show the roots of dynamic functions in functional programming and LISP. The next thing that John must implement to hook us completely, is execution on demand and thus infinite lists.

Eric Iverson – Java and J

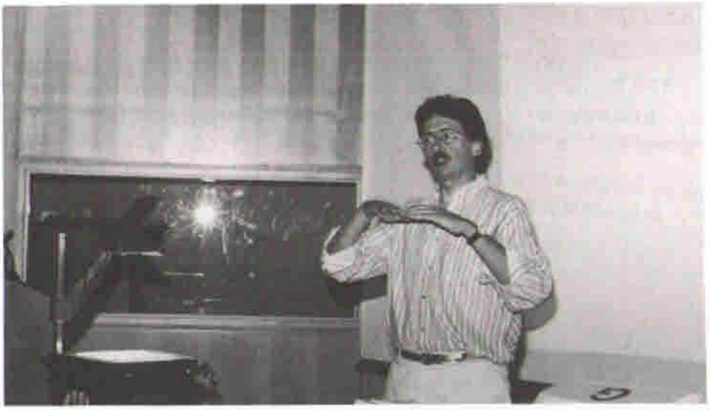
Eric's talk, falling as it did at the end of the conference, provided the cream on the bread and butter pudding. He gave a simple and immediately understandable explanation of the underlying internet technology, the best I have heard, and then went into what Java was all about. It is best perhaps to give some quotations.

On HTTP and HTML:

- "Like the best magic it relies on the simplest trick"
- "It is an OBPK" (which turned out to be Ian Sharp's One-Blade Pocket Knife)

On Java:

- "Java is not an OBPK. It has fork, spoon, kitchen sink, hot and cold running coffee — I LOVE it!"
- "It has all of the drawbacks of a compiled language and all the drawbacks of an interpreted language all rolled into one small neat package."
- "There are 100s and 1000s of Java books out there which are all the same apart from a few short sentences."



Eric Iverson on HTML

Having warmed the audience up suitably he could do no wrong and so he moved on to the main body of the talk. He demonstrated how to build a web page that took a J expression from a field and displayed the result of executing in another field. When the example worked he nearly got a standing ovation.

Conclusion

In summary it was a very worthwhile conference full of interesting talks. The facilities were adequate to the task and well worth the low cost of the conference. If they organised another I would probably be there.

Mostly on OpenGL

by Adrian Smith

Conference Roundup

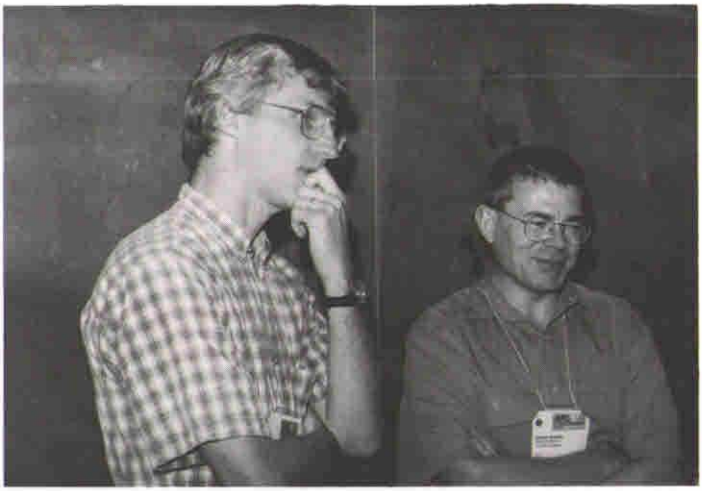
See above! This was the best APL conference I have been to, when rated on the yardsticks of value for money, facilities and organisation. It was packed with useful stuff, scheduled tightly over 3 days and run by the Toronto team with a careful eye on the budget and cash flow. They not only charged Vector $25 (the going rate) for the photos available 'over the counter' on the final afternoon, but offered to mail us the extras taken that day "for a few dollars more". I am sure they made money, and I hope secured the future of their APL interest group for many years to come — very well done!

The First-floor Landing at Ryerson during Coffee Break

I also enjoyed Ian Sharp's opening talk, and I find myself increasingly on the side of the Luddites when it comes to the APL2 extensions to his original one-bladed penknife that was VS APL. I think that if we had added rank, control structures and strings to the original definition back in 1976 we might still be up with Pascal and C++ as mass-market languages. I particularly appreciated the exchange between Ian and Jim Brown who pointed out that IBM had extended the penknife concept to something more general purpose, that would cut paper, twine, etc. etc. Ian added (almost *sotto voce*) "... and Sequoia trees" which pretty well sums it up for me.

Chris Burke on OpenGL Programming

Chris gave a very thorough introduction to the OpenGL concepts, and worked carefully through many of the sample scripts which are shipped with the J product. However he was careful to point out that the OpenGL calls are available to any Windows product, so there is no reason why you should not do all this stuff in APL+Win or Dyalog APL. All they have provided in J is a 'foreign conjunction' to streamline the interface.



Cliff Reiter with Chris Burke

Background to OpenGL

"Silicon Graphics" is a company which started as a purveyor of high-end special effects to the movie and advertising industry. They gradually accreted a library of functions for creating near-realistic images of geometric objects; initially these required enormously expensive and powerful Unix workstations to be effective, but the Intel platforms are catching up fast and so Silicon Graphics licensed the libraries to Microsoft for shipping with Win95 and NT. They are well demonstrated by several of the 32-bit screen-savers such as 'Pipes'.

J provides a very direct interface (Chris recommended that you buy the *OpenGL Programmer's Guide*, Addison Wesley, 0-201-46138-2) which is at the same level as that offered to C programmers but much easier to experiment with, as you can try things out and see the results on the fly. Naturally, J also hides boring things like data-type conversions so you can be that little bit less fussy about the exact function arguments.

Why Bother?

Chris gave three reasons:

(a) to have a lot of fun

(b) to render 3D graphics well. OK, you can do nice tiled mesh surfaces in J already, but these look blotchy without the artificial mesh, and they render poorly on monochrome devices (like printers). Higher resolution just adds processing and yields a finer degree of blotch! You also can never solve tricky problems like hiding the axes correctly when rendering a complex surface. Probably you just give up and box the entire graphic.

(c) Many fewer steps are required for smooth surfaces. OpenGL has splines built in, and the system defaults work well for you under most circumstances.

Interfaces at several levels

Chris started by showing the workings of OpenGL at the lowest level using functions like:

`glColor` — at least in J you just need one of these for all sensible arguments

`gluSphere` — things beginning `glu` are OpenGL utilities which bundle many commands into a single call

`glare` — a J interface addition to create a "render context"

So you can open the supplied script and experiment with simple line drawings such as:

```
glreset
wd Formdef  NB. a normal J window
glare
wd 'show;ontop'
glClearColor 0 0 1  NB. Red Green Blue (0-1)
glClear GL_COLOR_BUFFER_BIT
glSwapBuffers   NB Now it goes blue!
```

Question – what affects which buffer? Not absolutely clear, but updates seem to merge with the 'back' buffer, for example if we swop buffers again the window stays blue! Experimentation is probably the best course — as a last resort read the book!

```
NB. red triangle
....
glBegin GL_POLYGON
  glVertex 1 0 0
  glVertex 0 1 0
  glVertex _1 0 0
glEnd
```

As you can see, this involves learning a lot of new vocabulary and a script stacked with the kind of capitalised constants beloved of C programmers. However once you have made your simple shape you can get OpenGL to manipulate it.

Rotations can be nicely handled in J, particularly given the concept of function rank. You start with `glLoadIdentity` to set the current transformation to the unit matrix then use a 'rotation' function (rank-1) to make for easy XYZ rotations. Similarly you should define `glVertex` etc. as rank-1 so that they can take a matrix of point co-ordinates and iterate automatically over the rows.

Working with OpenGL Utilities

DisplayLists are a key component of OpenGL — effectively they store a preset sequence of canned commands which are partially compiled and execute much more efficiently than the same commands run over and over again. Probably you would use such a list for initialisation, then only change the elements that matter at each run. Chris demonstrated with a list to draw a dodecahedron, render the faces in random colours, light it from different viewpoints and so on.

Moving things around is where you can really scramble your brain! Chris started with the dodecahedron at the origin and showed how to move the viewpoint around a notional sphere at various distances from the object (of course you can also see it from the inside if you go in close enough). However you can translate and rotate the object too, and it sensibly spins about its local axes. As Chris amply demonstrated, if you try moving the viewpoint around at the same time as spinning and translating the object, you get some spectacular effects and completely lose any perception of which way is up!

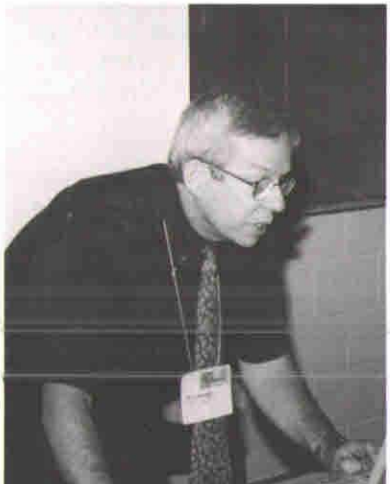
Summary

Obviously, most of what OpenGL does is visual, and notes like this fail to do it justice. I suppose the thing which has been bugging me since Toronto is "OK, but what would I use it for?". I don't make heavy use of spinning polyhedra, and if I need a photo-realistic image I can use a 35mm camera and a £600 scanner. I do need quality business and statistical graphics, but mostly I can represent the data better with a collection of simple 2D charts. The time taken to calculate and render anything other than the simplest scenes rules this technology out for any kind of interactive animation — such as allowing the user to spin your tower chart around — so we are left with the maths community who want to visualise the complex shapes that emerge from their equations. Is this really such a big advance from a good wireframe or open mesh surface? It would be great to have an opportunity to play with this stuff, but to hark back to Chris's initial set of bullet points, the only honest answer to why bother would be *"(a) to have a lot of fun"*. I would love to be proved wrong.

A Flavour of APL97

photographs by Richard Proctor

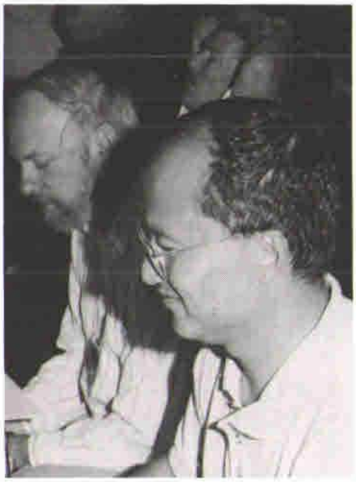
Workshops

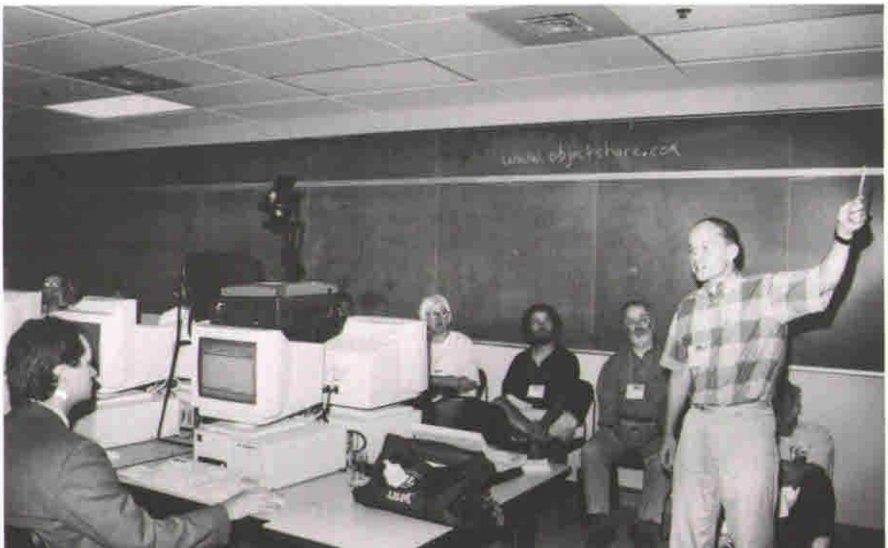

Eric Lescasse on Object-oriented Programming

Chris Lee on User-defined Classes in APL+Win

More Workshops ...

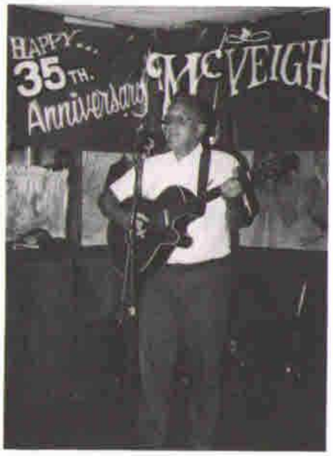
Chris Lincoln & Son


Richard Brown, Roger Hui with Jim Lucas in the background



Timo Laurmaa with his Web server

Relaxing ...

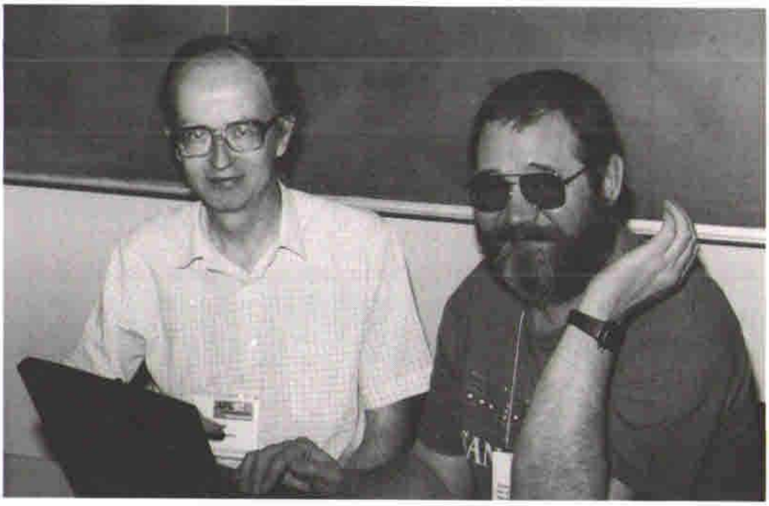


Jim Brown plays Guitar

Alan Graham also ...

COSY in the Snug at McVeighs

Time to do some Real Work ...


Adrian Smith and Sasha Skomorokov working on AP205

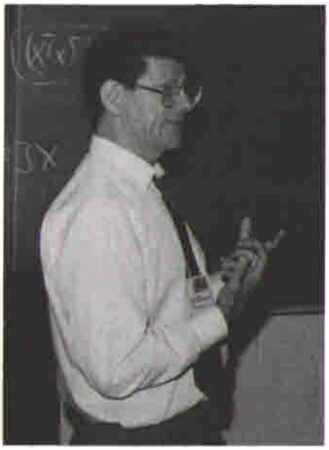
The Adaytum Team Reflecting

APL and Nested Arrays - a Dream for Statistical Computation

by Alan Sykes and Tom Stroud

Introduction

Many papers have been produced in the last 10 or more years in the APL community extolling the virtues of APL for statistical computing so why another one? Such papers have rightly stressed APL's array-handling operations, the use of user-defined operators, the often transparent flow from mathematical notation to APL's equivalent computational notation, together with specific tools with a statistical bent, like the dyadic domino.


Alan Sykes

Nested arrays however figure relatively little in this. Of course the current Windows-oriented computing medium means that a function's argument could be, for example, a list of properties of an object. For example drawing a polygonal line in Dyalog APL requires an incantation such as:

```
'F.L1'⎕wc'Poly' (Y X)(0 0 0)('Lwidth' 5)
('Fstyle'0)('Fillcol' 255 0 0)
```

Nested arrays play a crucial role here. Similarly, our use of nested arrays in APL programming tends only to exploit nested array structures for passing complex arrays in and out of user-defined functions.

However, most of statistics involves number crunching, calculating means and standard deviations. This paper discusses a particular statistical application for which nested numeric vectors and matrices are a natural data structure for which means and variances need to be calculated. This application is **Multiple Imputation**.

Because we will be dealing with numeric vectors or matrices, which have at least one scalar, which when disclosed, is itself a numeric vector, the principle of scalar extension will play a fundamental role in what follows. The following code illustrates this simple idea.

```
      +/1 2 3 (1 2 3)
 7 8 9
```

Introducing Multiple Imputation

Among the greatest users of statistical techniques are the Market Researchers who often collect information about customers through a survey. Social scientists have used surveys too, many of them huge, to investigate specific social scientific questions relevant to modern day life. The bugbear of all such survey work, no matter how much work is put into preparing the right kind of questions and the best survey design, is non-response. Multiple Imputation [1] is, in the words of its originator (D. Rubin), *the technique that replaces each missing value with two or more acceptable values representing a distribution of possibilities.*

The need for it stems from the fact that:

- to ignore item non-response is potentially disastrous for subsequent inferences made from the database;
- if single imputation is used, then complete-data methods of analysis may be used and the need for specialist computer programs to handle non-response is avoided;
- single imputation, however well chosen, means that inferences based on the imputed data will be too sharp since the extra variability due to the unknown missing values is not being taken into account.

Multiple imputation proceeds by:

- imputing a modest number *m* (between 2 and 10) missing values by some appropriate means;
- producing the required statistical analysis using all *m* different databases;
- using the variation in the *m* different results to tease out the extra variability due to the unknown missing values.

APL to the Rescue

With this scenario in mind, one can begin to see the role APL can play in multiple imputation calculations. Rubin suggests for example that the imputed values 'are stored in an auxiliary matrix with one row for each missing value and *m* columns'

This implies straightaway that specialist programs will have to be constructed. For example, calculating a simple mean of a variable that has been imputed, would presumably require a program that loops round each imputation, which it would have to know where to find, in order to produce the *m* means.

In APL, we do not need any auxiliary matrix, nor do we need a specialist program - an appropriately ordinary *MEAN* program should suffice! Take a simple example. We have a two-variable database and the Y-variable (first column) has two missing values which have been imputed *m*=2 times:

	Y	X
1	10	8
2	?	9
3	14	11
4	?	13
5	16	16
6	15	18
7	20	6
8	4	4
9	18	20
10	22	25

	Repetition One	Repetition Two
Unit 2	10	14
Unit 4	16	14

The important point to realise is that we have **two databases** here - one with each unknown value replaced by 10 and 16 respectively, and one with them replaced by 14 and 14. Thus if we require the mean of the Y column we must do one calculation substituting the values 10 and 16 for the two queries, and a further calculation substituting the values 14 and 14. In APL, the original database plus the auxiliary matrix of imputed values may be stored as one matrix and means can be calculated directly:

```
      data←10 2⍴10 8 0 9 14 11 0 13 16 16 15 18 20 6 4 4 18 20
 22 25
      data[2;1]←⊂10 14
      data[4;1]←⊂16 14
      data
     10    8
  10 14    9
     14   11
  16 14   13
     16   16
     15   18
     20    6
      4    4
     18   20
     22   25
```

```
      R←MEAN X
[1]   R←(+/X)÷θρρX

      MEAN data[;1]
 14.5 14.7

      MEAN data
 14.5 14.7  13

      DISP MEAN data[;1]
┌─────────┐
│14.5 14.7│
└─────────┘

      DISP MEAN data
┌─────────┬──┐
│14.5 14.7│13│
└─────────┴──┘
```

As you can see, this works superbly, irrespective of whether we require just the mean of the y-data or require means of both columns. The only proviso, from an APL point of view, is that the number of imputations is the same for each missing value. But this is also required from a statistical point of view so there is no restriction in its application. In a similar fashion, many routine statistical calculations in APL translate to the nested-array scenario painlessly. For example, take the functions *SSQ* (Sum of squares), *SPR* (sum of products), *SDEV* (Standard deviation) given by:

```
        R←SSQ X;T
[1]     R←T+.×T←X-MEAN X

        R←X SPR Y
[1]     R←(X-MEAN X)+.×Y-MEAN Y

        R←SDEV X;T
[1]     R←((+/T×T←X-(ρX)ρMEAN X)÷¯1+θρρX)*0.5
```

It is easy to see that these simple functions (that our students would write as part of their first assignment in APL Statistical Computing) do what we want, so we can use them to calculate the slope and intercept of a regression line of *Y* on *X*:

Regression Line Slope:

```
      ⎕←B←(X SPR Y)÷SSQ X
 0.5447761194 0.5049751244
```

Regression Line Intercept:

```
      ⎕←A←(MEAN Y)-B×MEAN X
7.417910448 8.135323383
```

In each case the returning result is an enclosed vector of length 2 because the number of imputations is 2.

Of course, advanced functions of the APL interpreter (e.g. ⌹) will not operate on nested arrays. Nevertheless, as we show in the next section, it is possible to write simple first-generation APL code in such a way that the dyadic use of ⌹ can be achieved, and in addition, **that same code will work quite naturally for nested arrays!**

APL Code for Dyadic Domino

The most common statistical use for the Dyadic Domino is to calculate the least-squares regression coefficients in a multiple linear regression problem. If *Y1* is the vector of 'y-values' taken from the first copy of *Y* and *D* is the design matrix of linear predictors, i.e.

```
      D←1,[1.5]X←8 9 11 13 16 18 6 4 20 25
      D
1  8
1  9
1 11
1 13
1 16
1 18
1  6
1  4
1 20
1 25
```

then we need the following important calculations:

- the least-squares estimates of the coefficients of the linear predictors;

```
      B←Y1⌹D
```

- the residual Sum of Squares;

```
      RSS←+/T×T←Y1-D+.×B
```

- the covariance matrix of the regression estimates;

```
      COV←(RSS÷-/⍴D)×⌹(⍉D)+.×D

      B
7.417910448 0.5447761194

      RSS
135.1940299

      COV
 8.794338011   ¯0.5464932799
¯0.5464932799  0.04203794461
```

(Note that since `Y1` is the first copy of `Y` then the values of `B` equal the first values of the slope and intercept previously calculated.)

To have this facility for a general design matrix `D`, which together with `Y` may contain enclosed vectors of the same length, we appeal to the efficient and elegantly designed Beaton's algorithm (see [2]) which is similar to Gaussian elimination. It can be coded easily in APL and the coding works **without alteration** for nested arrays.

```
      R←Y LS X;FL;X;Y;Z;K;TV;T;XTX;B;SS;D;A;B0
[1]    K←1 ⋄ Z←X,Y ⋄ Z←(⍉Z)+.×Z ⋄ D←⍬⍴⍴Z
[2]   ⍝ do Beaton recipe on sums of squares matrix
[3]   ⍝ NB - at the moment it does nothing about checking whether to
[4]   ⍝ cope for ill-conditioning if the first column in design
[5]   ⍝ matrix is all 1's - it pivots on all columns!
[6]   LP:Z←Z-(TV∘.×TV←Z[K;])÷T←Z[K;K]
[7]    Z[;K]←Z[K;]←TV÷|T
[8]    Z[K;K]←-÷T
[9]    →((¯1+1 ⍴Z)≥K←K+1)/LP
[10]  ⍝ now allocate separate results for clarity
[11]   XTX←-¯1 0↓0 ¯1↓Z ⋄ SS←Z[D;D] ⋄ B←¯1↓Z[;D]
[12]   R←B SS XTX
```

We use nested arrays **only** to return the three arrays as one, following the specification above – i.e. the first element consists of the regression coefficients, the second equals the residual sum of squares, and the third equals the estimated covariance matrix of the regression coefficients. Again, we illustrate it using `Y1` equal to the vector of first y-values in `Y` and also with `Y2` as the vector of the second y-values in `Y`, suppressing all but two decimal places in the output.

```
      DISP 2 DP Y1 LS D
┌─────────┬──────┬──────────┐
│7.42 0.54│135.19│ 8.79 ¯0.55│
│         │      │¯0.55  0.04│
└─────────┴──────┴──────────┘
```

Note that the first component gives the intercept and slope values, the next is the regression sum of squares and finally we have the 2 by 2 covariance matrix. Similarly for `Y2`:

```
      DISP 2 DP Y2 LS D
┌────────┬──────┬──────────┐
│8.14 0.5│129.59│ 8.43 ¯0.52│
│        │      │¯0.52  0.04│
└────────┴──────┴──────────┘
```

The same code, containing no second generation idioms apart from the output of the three arrays now works for the nested `Y` vector:

```
    DISP 2 DP Y LS D
┌───────────────────┬───────────────┬───────────────────────┐
│                   │               │┌──────────┬──────────┐│
│┌─────────┬────────┐│┌─────────────┐││ 8.79 8.43│¯0.55 ¯0.52││
││7.42 8.14│0.54 0.5│││135.19 129.59│││──────────┼──────────┤│
│└─────────┴────────┘│└─────────────┘││¯0.55 ¯0.52│ 0.04 0.04││
│                   │               │└──────────┴──────────┘│
└───────────────────┴───────────────┴───────────────────────┘
```

Note that the 2 by 2 covariance matrix is nested because there are two residual sum of squares and **not** because the design matrix is nested. However, there is no reason, either theoretical or practical why the design matrix should not be nested. Consider replacing the value `25` in the `[10;2]` position of `D` by the enclosed vector `25 30`:

```
      D[10;2]←⊂25 30
      DISP 2 DP Y LS D
```

```
┌────────────────────┬───────────────┬───────────────────────┐
│                    │               │┌──────────┬──────────┐│
│┌─────────┬─────────┐│┌─────────────┐││ 8.79 6.89│¯0.55 ¯0.39││
││7.42 8.76│0.54 0.44│││135.19 126.76│││──────────┼──────────┤│
│└─────────┴─────────┘│└─────────────┘││¯0.55 ¯0.39│ 0.04 0.03││
│                    │               │└──────────┴──────────┘│
└────────────────────┴───────────────┴───────────────────────┘
```

As is to be expected, all first values of each element are the same as before, whereas second values are different, as for those results the 10th x-value is now `30` and not `25`.

Generalized Linear Models

As is well documented in previous APL papers, (see e.g.[3]) it is possible using an iteratively re-weighted least-squares procedure to extend the range of validity of regression-type problems to situations where the 'y-data' is not normally distributed. A typical example might be where each y-value represents the number of successes out of a number of trials and where the aim is to explain how the probability of success varies with the predictor values. By a direct emulation of the software of *ASLGREG* (see [5]) for this particular example of a generalized linear model, we can achieve estimates for a 'logistic regression' problem.

```
     RES←Y LOGREG X;T;MU;TP;P;N;ETA;DETA;C;WT;WV;BETA;V;D
       ⍎(1=⍴⍴Y)/'N←(⍴Y)⍴1'
       ⍎(1≠⍴⍴Y)/'N←Y[;2]⋄Y←Y[;1]'
      ⍝start
       MU←Y ⋄ C←1 ⋄ DEV←⌊/⍳0 ⋄ P←¯1 ⍴X
       ETA←⍟TP÷1-TP←(TP+0.0001×TP=0)÷N+N=TP←MU
      ⍝ now start iteration
      LP:DETA←((1+D)*2)÷N×D←*ETA
       WV←ETA+(Y-MU)×DETA
       WT←N×TP×1-TP
       WT←WT*0.5
       BETA←1⊃BT←(WT×WV)LS X×(⍴X)⍴P/WT
       ETA←X+.×BETA ⋄ MU←N×TP←D÷1+D←*ETA
       DEVNEW←+/2×(Y×⍟(Y+Y=0)÷MU+0.001×MU=0)+(N-Y)×⍟(N-Y
       -Y=N)÷(N-MU)+0.001×0=N-MU
       ⍎(0.00001<⌈/|DEV-⊃DEVNEW)/'C←C+1⋄DEV←⊃DEVNEW⋄→LP'
       RES←BETA DEVNEW((3⊃BT)×(-/⍴X)÷2⊃BT)
```

The left argument is either a vector of 0s and 1s (successes out of 1 trial) or a matrix whose first column contains the number of successes and the second column the corresponding total number of trials. The right argument is a matrix of predictor variables. A simple example shows how both can be nested.

```
      DISP Y
┌───┬─┬─┬─┬─┬─┬─┬─┬─┬─┐
│0 1│0│1│0│1│1│0│1│1│1│
└───┴─┴─┴─┴─┴─┴─┴─┴─┴─┘
      X←1,[1.5]⍳10
      X[1;2]←⊂1 5
      DISP X
┌─┬───┐
│1│1 5│
├─┼───┤
│1│ 2 │
├─┼───┤
│1│ 3 │
├─┼───┤
│1│ 4 │
├─┼───┤
│1│ 5 │
├─┼───┤
│1│ 6 │
├─┼───┤
│1│ 7 │
├─┼───┤
│1│ 8 │
├─┼───┤
│1│ 9 │
├─┼───┤
│1│10 │
└─┴───┘

      DISP 3 DP Y LOGREG X
┌─────────────────────────────┬──────────────┬──────────────────────────────┐
│                             │              │┌─────────────┬──────────────┐│
│┌───────────┬───────────┐    │┌───────────┐ ││ 3.233 3.903 │¯0.545 ¯0.671 ││
││¯2.29 ¯1.548│0.528 0.444│   ││9.803 10.32│ │├─────────────┼──────────────┤│
│└───────────┴───────────┘    │└───────────┘ ││¯0.545 ¯0.671│ 0.114 0.136  ││
│                             │              │└─────────────┴──────────────┘│
└─────────────────────────────┴──────────────┴──────────────────────────────┘
```

As before, the first element of the result is the nested vector of 'regression coefficients'. The second vector is now the 'deviance' (equivalent to the residual sum of squares) and the final element is the covariance matrix of the regression estimates. These results may be verified by repeated use of *ASLGREG*. (Note however that accuracy of the results is not as high as it should be - see the comments in the function *LS*.)

Other Utilities

To finish with we look at some other programming issues associated with the use of nested arrays for multiple imputation. (It is hoped to write up the details of an actual application of multiple imputation elsewhere as they are beyond the scope of this paper.)

Identifying rows of a nested matrix

Suppose *X* is a nested matrix (in this case with two columns). We wish to identify each row in terms of its position in a set of unique rows (given by the columns of *T*). Standard first-generation look-up code works wonderfully once we have found *T*. For this, we first have to convert *X* to a non-nested version which includes all possible rows:

```
      DISP X
┌───┬───┐
│1 2│2  │
├───┼───┤
│ 1 │ 1 │
├───┼───┤
│ 2 │2 1│
├───┼───┤
│ 1 │ 2 │
└───┴───┘

      ⊃,/(,¨¨)/X
1 2
2 2
1 1
2 2
2 1
1 2
```

Then a simple *UNIQUE* function which identifies unique elements of a vector or unique rows of a matrix will do the job:

```
      UNIQUE 2 1 1 2 2 3
1 2 3
      UNIQUE 3 2⍴1 2 2 1 1 2
1 2
2 1

      ⎕←T← ⍉UNIQUE ⊃,/(,¨¨)/X
1 1 2 2
1 2 1 2
```

Now we can identify which rows of the original (nested) *X* matrix are which:

```
      DISP (X∧.=T)+.×⍳4
┌───┬─┬───┬─┐
│2 4│1│4 3│2│
└───┴─┴───┴─┘
```

The interpretation of this is clear – the first element which is nested tells us that the first row of *X* (which is `1 2` or `2 2` depending on which copy we take) consists of the 2nd or 4th elements in the unique listing. Similarly for all the other elements.

So indexing the rows of a nested matrix requires converting the nested matrix to a non-nested matrix. Calculating the median of each column of a matrix requires a similar conversion. For example, the function *NVTOMAT* (*NMTOMAT*) converts a nested vector (matrix) to a matrix.

```
      ∇ r←NVTOMAT v;m
        ⍝ converts nested vector to matrix
         m←⌈/,⊃¨ρ¨v
         r←↑mρ¨v
      ∇

      ∇NMTOMAT∇
      ∇ r←NMTOMAT mat;m;d
        ⍝ converts nested matrix to fatter matrix
         d←ρmat
         m←⊃¨ρ¨mat ⋄ m←dρ⌈⌿m+m=0
         r←(d[1],+/m[1;])ρ⊃,/,mρ¨mat
      ∇
```

To illustrate these functions consider the nested vector v and the nested matrix *m* below:

```
      DISP v
┌─────┬─┬──┐
│1 2 3│9│99│
└─────┴─┴──┘

      NVTOMAT v
 1  2  3
 9  9  9
99 99 99

      DISP m
┌─────┬─┐
│1 2 3│2│
├─────┼─┤
│  3  │4│
├─────┼─┤
│  5  │6│
└─────┴─┘

      NMTOMAT m
1 2 3 2
3 3 3 4
5 5 5 6
```

Calculating Medians

Norman Thomson in his St Petersburg paper ([4]) refers to the median function as a function which is 'not dimensionally extendible' by which he is suggesting that (unlike the *MEAN* program) it is not possible to program *MEDIAN* so that it works on an array of any dimension without looping. This is not true as the following program illustrates.

```
      R←MEDIAN X;ORD;ND;IND;D
        D←⍴X
        ⍎(1=⍴D)/'X←((⍴X),1)⍴X' ⋄ ND←⍴X
        IND←ND⍴⍳ND[2] ⋄ X←,⍉X ⋄ IND←,⍉IND
        X←X[ORD←⍋X] ⋄ IND←IND[ORD]
        X←X[⍋IND]
        X←⍉(⌽ND)⍴X
        R←0.5+.×X[⌈0.5×0 1+D[1];]
        ⍎(1=⍴D)/'R←⍬⍴R'

      D
1  8
1  9
1 11
1 13
1 16
1 18
1  6
1  4
1 20
1 25

      MEDIAN D
1 12
```

If we wish to extend the usefulness of the program *MEDIAN* to nested arrays of the type encountered in multiple imputation, then there is no real way of avoiding converting the nested vector to a matrix, use *MEDIAN* to calculate the median value for each column, and then convert this back to have the appropriate nested structure.

```
      DISP Y
┌──┬─────┬──┬─────┬──┬──┬──┬─┬──┬──┐
│10│10 14│14│16 14│16│15│20│4│18│22│
└──┴─────┴──┴─────┴──┴──┴──┴─┴──┴──┘

      IMEDIAN Y
15.5 14.5
```

```
      DISP IMEDIAN D,Y
┌─┬──┬─────────┐
│1│12│15.5 14.5│
└─┴──┴─────────┘

      r←IMEDIAN arr;i;m;k
       ⍝computes median of nested vector/matrix
        ⍎(1≥|≡arr)/'r←MEDIAN arr⋄→0'
        →(vec,mat)[⍴⍴arr]
       vec:r←MEDIAN NVTOMAT arr ⋄ →0
        mat:m←⌈⌿m←⊃¨⍴¨arr ⋄
        m←m+m=0
        m←⊃,/m ¨1
        r←m⊂MEDIAN NMTOMAT arr
```

Conclusion

Nested numeric arrays in APL form a natural tool for use in database work involving multiply imputed values. We have been pleasantly surprised how much basic APL code works with the data structures discussed in this paper and how quickly other facilities can be constructed. The simple idea of scalar extension at the heart of both first- and second-generation APL ensures that the statistician can largely operate with nested arrays in the sure knowledge that the results will be what they obviously should be with nested elements containing the results corresponding to each choice of elements in the nested elements of the original array. In short, the illustration of the *MEAN* program operating on a nested vector or matrix says it all!

References

[1] Rubin, D.B. (1987), *Multiple Imputation for Nonresponse in Surveys*. Wiley, New York.

[2] Sykes A.M. and Ansell J.L., *Beaton's Recipe for Least-Squares*. Vector Vol. 8, No. 1, July 1991, pp36–44.

[3] Sykes A.M., *Generalized Linear Models in APL*, Proceedings of APL86 (Manchester) pp 48–70.

[4] Thomson N., *Some Proposals for APL2 Specification of Statistical Algorithms*, APL92 Conference Proceedings (St Petersburg), APL Quote Quad Volume 23 Number 1, July 1992. pp 243–255.

[5] Morgan E., *ASLGREG: Proposed Additional Features, Predictions with Confidence Limits*, Vector Vol. 10 No. 3, January 1994, pp 81–93.

Interactive Design of Structures: A Program for Everyone

by Johann Riebenbauer and Joachim Hoffmann

Abstract

In this paper we present a program for the design and analysis of planar member frame structures. The intuitive graphic user interface covers all steps necessary for designing a load bearing structure, from geometry design to system optimisation.

This project is the first step in converting from a pool of APL*PLUS II programs towards interactive Windows applications written in *DyalogAPL/W*, and was carried out by Dipl.-Ing. Johann Riebenbauer, Institute of Structural Design (Institut für Tragwerkslehre) at the Technical University of Graz, Austria, and the Group of Civil Engineering (IG für Bauwesen) Zenkner&Handel.

The computational section of the described application is composed of proven space frame algorithms, which where developed by Prof. Peter Klement over the last three decades in APL, and which represent the backbone of the new graphic user interface (GUI) in *DyalogAPL/W*.

(In the Appendix the reader can find a small vocabulary, which can be used to translate the German words in the illustrations.)

Interactive Design of Structures

Introduction

This software for the design and analysis of planar member frame structures was the central part of Johann Riebenbauer's MSc thesis, awarded by the Institute of Wooden Constructions (Institut für Holzbau) of the TU Graz. The main emphasis was not on the determination of the internal forces of members or deformations, but on the organisation and handling of data before and after the calculation (pre- and post-processing of the system). Restriction to planar structures was only imposed by the GUI, all the computational functions in the background are designed to work with 3D-frameworks. Future versions of the GUI will also work with space frame structures and area covering structures.

During program design, priority was given to the development of a single interactive and intuitive application-GUI and not a conventional program, which usually consists of three parts, namely preprocessor, calculation processor and finally postprocessor.

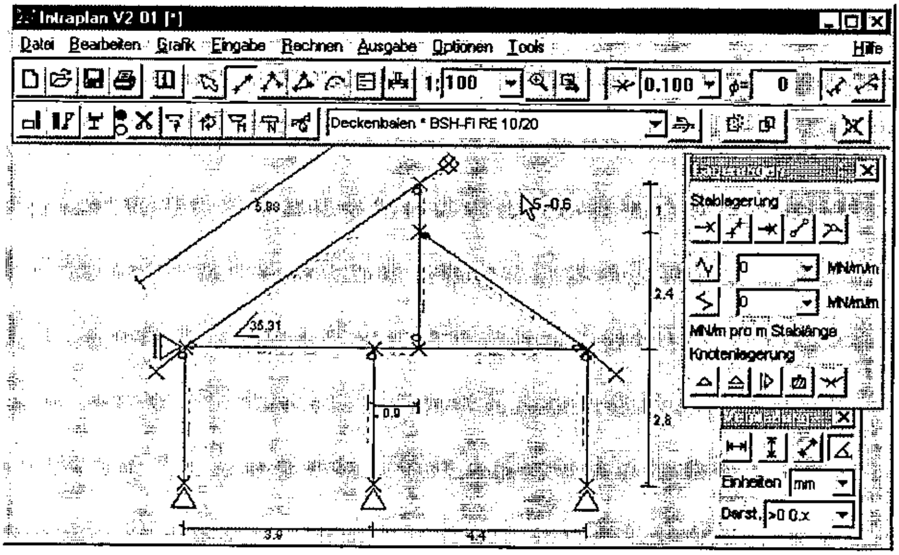


Figure 1: Display of a system (a house) with supports and measurements

Each action of the user is displayed immediately on the screen in a clear fashion – following the Windows philosophy "What You See Is What You Get". Input errors are verified immediately, and not only after computations or in the form of a system crash. Special errors are documented with informative error messages. Help and guidance is also provided by a standard Windows help file.

Motivation

The various processes when dealing with structural problems, e.g. system geometry design, assumed load design, dimensioning, calculation of system forces, interpretation of results, design of individual frame elements, presentation of significant results for further processing, etc. are more or less automated in the everyday life of a civil engineer. With the development of new European Standards (Euro-Codes) continuous automation of all processes will be increasingly important.

This program will only be used in practice by constructing engineers (e.g. for small problems like a single-span beam), if it is simple and user-friendly.

The GUI takes into account various habits of engineers, but it does not impose a fixed way of working on them. Automation must not restrict, but has to support the working process at maximum. Therefore this GUI is designed similarly to the manual calculation on paper and follows the concepts of modern text-processors or spreadsheet programs.

System Design

The input of a system with all its definitions of hinges, supports is the most important, but also a highly time consuming part of the analysis of structures. Here, personal experience plays a major role. Also, checking the correctness of data input is of significant importance.

A CAD application could be considered a solution, but there is a significant difference between engineering a load bearing structure and drawing a sketch.

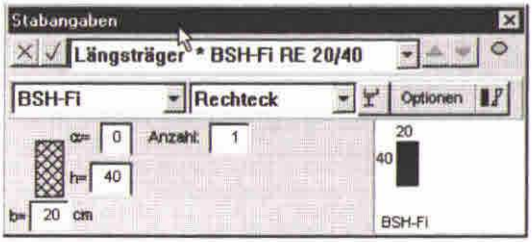
This GUI is developed specially for the input and handling of structural systems. The possibility of drawing a system by hand (with the mouse) makes it very easy for a non-CAD designer to design his member frame quickly. Structures can be designed without entering any numbers. With only a few, but effective operations, frameworks can be designed in a fashion similar to drawing on a sheet of paper by hand. Traditional preparations, e.g. system sketch and co-ordinate tables, are superfluous.

The structure on the screen can be changed in any way, e.g. translated, copied, scaled, deleted, etc., whereupon all relevant properties of nodes and members are changed automatically. Individual members and nodes can be marked with the mouse and can be assigned properties, which are again immediately displayed in graphical form. This makes it simple to detect input errors when they occur.

For the construction of circular arcs, parabolic curves, trussed girders, or simply for partitioning a member into equal parts, node-generation-aids are provided. These facilities can be extended to more complex structure-generators by the user.

Specification of Cross-Sections

For a realistic design of structures, which normally consists of different materials and member cross-sections, a separate dialogue box is provided. In this dialogue, all the relevant materials and cross-sections can be specified and given names similarly to the format-style concept in popular text processors. These specifications can then be assigned to individual members or groups of members by simply selecting them with the mouse and choosing a "member-style-name" from the member specifications dialogue. To distinguish different cross-sections, upper surfaces, lower surface and different materials can be displayed in different colours. The cross-section itself is also displayed in correct scale in the member specification dialogue (please see the form labelled with "Stabangaben"). In addition, predefined cross-sections (HEA, HEB, etc.) can be used.


Stabangaben

Längsträger * BSH-Fi RE 20/40

BSH-Fi | Rechteck | Optionen

α= 0 Anzahl: 1

h= 40

b= 20 cm

Figure 2: Member specifications with outlined cross-sections

The cross-section values are converted internally to a single reference material. There is no need to calculate moments of inertia or areas by hand.

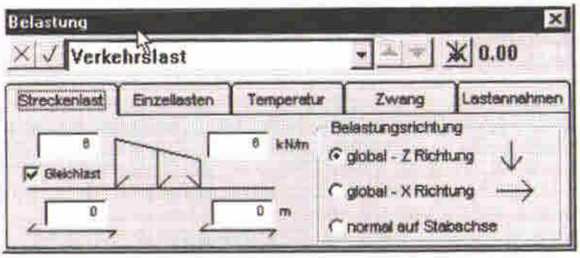
The integration of QS-WIN is planned for a later version. QS-WIN is a separate (DyalogAPL/W) program for the calculation of the properties of cross-sections with general outline and dimensions.

Loading the system

The process of loading the structure has been carefully analysed and automated. Due to the new and complex CEN standards effective automation has become necessary.

Each load case is given a name, which will be used in further processing. Recurrent and composed load cases can be produced automatically, e.g. the min-max-superpostion of different traffic load cases (where the single load case is not of interest) is defined under a single name, and the calculation and superposition of the subordinate load cases is done automatically by the program.

This also applies to snow and wind stresses. In the case of snow, even asymmetric position of loads are taken into account.


Belastung

Verkehrslast 0.00

Streckenlast | Einzellasten | Temperatur | Zwang | Lastannahmen

8 — 8 kN/m

[✓ Gleichlast]

0 — 0 m

Belastungsrichtung

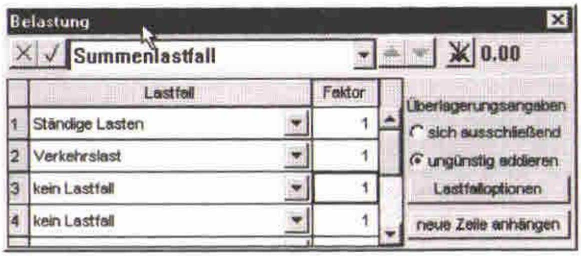
[✓ global - Z Richtung]

global - X Richtung

normal auf Stabachse

Figure 3: Definition of different load cases

Special algorithms for crane loads and traffic loads have been developed. The algorithms solve these problems by defining load groups or load bands. This simplifies everyday work significantly. The definition and assignment of stresses to individual members or nodes works similarly to the member specification dialogue. The user marks several members or nodes and puts the load onto them by selecting a load name from the load case dialogue.


Belastung

Summenlastfall 0.00

	Lastfall	Faktor
1	Ständige Lasten	1
2	Verkehrslast	1
3	kein Lastfall	1
4	kein Lastfall	1

Überlagerungsangaben

sich ausschließend

[✓ ungünstig addieren]

Lastfalloptionen

neue Zeile anhängen

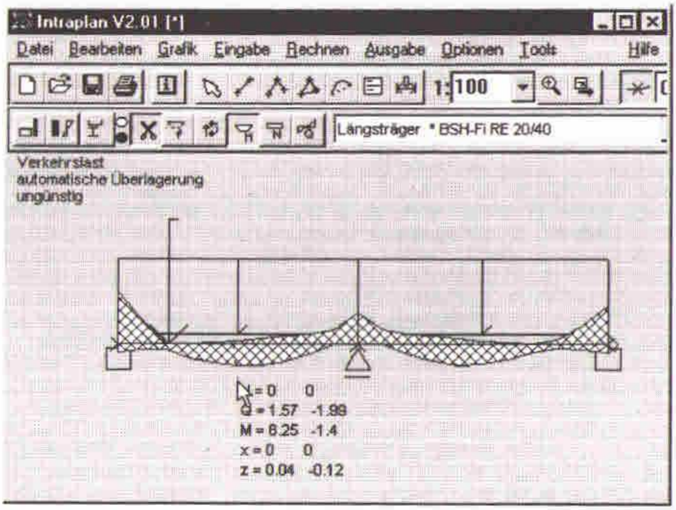
Figure 4: Definition of load case superposition

Of course, superimposing individual load cases is possible. Also, the prerequisites to perform calculations following the European structural standards (Euro-Codes), which requires the superpositions of a variety of load cases, are provided. These automated options are of special interest, because they make the many load cases required in the Euro-Codes easy to manage.

Display of Results

Internally the separate load-cases and corresponding internal forces are superimposed, however on the screen only significant min-max-values of different types of internal forces are displayed. This is necessary for a clear display of results. In most cases such an overview, displaying the quality of results is sufficient in practice.

If more exact quantities of a result are needed (e.g. absolute values of a deformation curve), a scissors function is provided. The user simply cuts an arbitrary element of the structure with a scissors cursor and the values are displayed on the screen. Furthermore, a similar function exists to determine the stresses and internal forces of a selected cross-section, which serves to check the former cross-section dimensioning.


Intraplan V2.01 [*]
Datei Bearbeiten Grafik Eingabe Rechnen Ausgabe Optionen Tools Hilfe
1:100
Längsträger * BSH-Fi RE 20/40

```
Verkehrslast
automatische Überlagerung
ungünstig
```

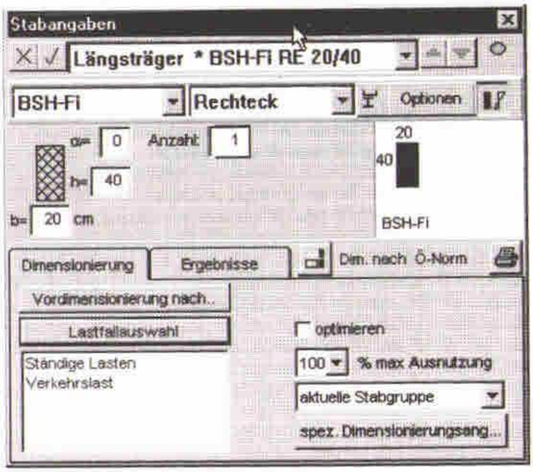
```
N = 0     0
Q = 1.57  -1.98
M = 6.25  -1.4
x = 0     0
z = 0.04  -0.12
```

Figure 5: Min-max superimposed moment curve

Internal forces can be displayed not only for selected members, but also for predefined member groups. With a standard zoom function special areas can be viewed in detail. The values of support reactions and elastic support forces can be displayed and measured with the mouse (see Fig. 5).

Design of Individual Structure Elements

Of course, it is possible to design and optimise individual cross-sections in detail, according to various standards. Using the "Extended Member Specification Dialogue" (See Fig. 6), the following properties of cross-sections can be defined: load factor, planar and 3D-buckling length (at the moment following only the equivalent beam method), variable girder depth, special superposition specifications, bent girder axis, etc.



Längsträger * BSH-Fi RE 20/40

BSH-Fi | Rechteck

α= 0 Anzahl: 1

h= 40

b= 20 cm

20 40 BSH-Fi

Ständige Lasten
Verkehrslast

optimieren

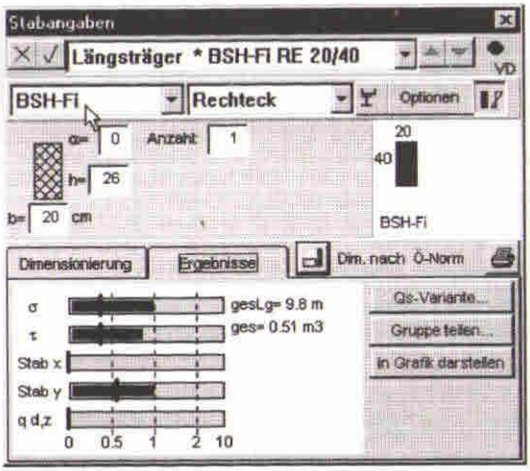
100 % max Ausnutzung

aktuelle Stabgruppe

Figure 6: Dimensioning specifications

The next important step is the interpretation of results and the possibility to alter design specifications in a very simple way, in order to optimise the structure. This applies to system geometry, cross-sections, supports and subdivisions of members. The design results are visualised by showing the load factors of all member groups (see Fig. 7) and the curves of load factors over the members themselves (see Fig. 8).

Furthermore the average load factors weighted over the length of the members is displayed, so that it is convenient to judge load factors of whole member groups.



Längsträger * BSH-Fi RE 20/40

BSH-Fi | Rechteck

α= 0 Anzahl: 1

h= 26

b= 20 cm

20 40 BSH-Fi

ges.Lg= 9.8 m

ges= 0.51 m3

Figure 7: Load-factors of the selected members

As mentioned above, it is also possible to dimension cross-sections by specifying internal forces, buckling lengths and member lengths.

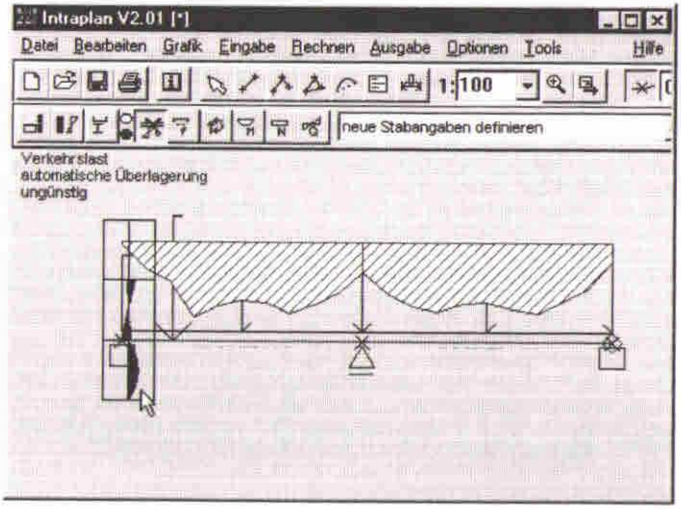

Intraplan V2.01 [*]
Datei Bearbeiten Grafik Eingabe Rechnen Ausgabe Optionen Tools Hilfe
1:100
neue Stabangaben definieren
Verkehrslast
automatische Überlagerung
ungünstig

Figure 8: Load-factors of the max-loaded section

In the system graphic, the distribution of load factors is displayed over the member lengths. A large crosshatched area over a member indicates that this member is not stressed to its capacity (Fig. 8). Ineffectively used members are therefore detected immediately. The program automatically suggests optimised cross-section specifications, displayed in the relevant edit windows, in order that the user can accept them directly with one keystroke, but can also change them as required.

Finally all the results can be printed in graphical and numerical form.

Final Remarks

The aim of this program is to provide an intuitive graphic user interface for the complete treatment of load bearing structures, starting from defining the geometry and loads leading to a clear presentation of results on the screen and on the printer. In the near future, a Structure Viewer will be released, which will only display and print existing data from file, so that structures can be interchanged electronically.

Acknowledgments

Peter Klement, Prof. Emeritus at the Institute of Structural Analysis (Institut für Baustatik) at the Technical University of Graz, Austria, is a well known structural engineer in Europe and APL pioneer in Austria.

IG Zenkner&Handel is a specialist civil engineering company, which specialises in structural calculations, e.g. dynamic analysis, huge roof constructions with large deflections, etc.

Dr. Günther Zenkner and Dr. Erich Handel are constantly developing and marketing several structural analysis software packages written in APL*PLUS II and DyalogAPL/W, whose theoretical foundations were laid during ten years of research work as research assistants at the Institute of Structural Analysis under Prof. Klement. *Intraplan* is under copyright of IG Zenkner&Handel.

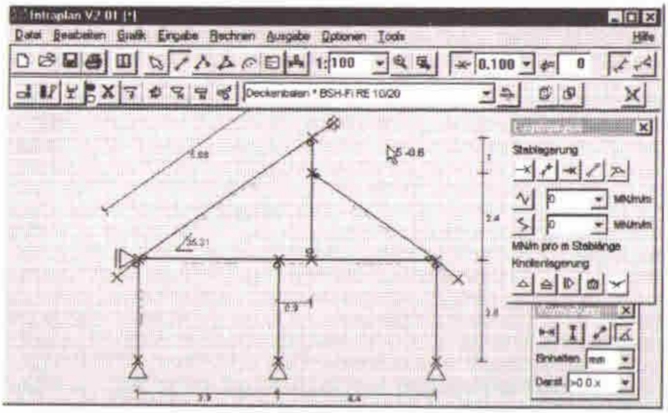
The Authors

Dipl.-Ing. Johann Riebenbauer is a research assistant in the department of Structural Design at the TU Graz, Austria. He is working on the research and development of constructive timber components of wooden frame structures and he is implementing his research results as DyalogAPL programs, which are ready for use in practice.

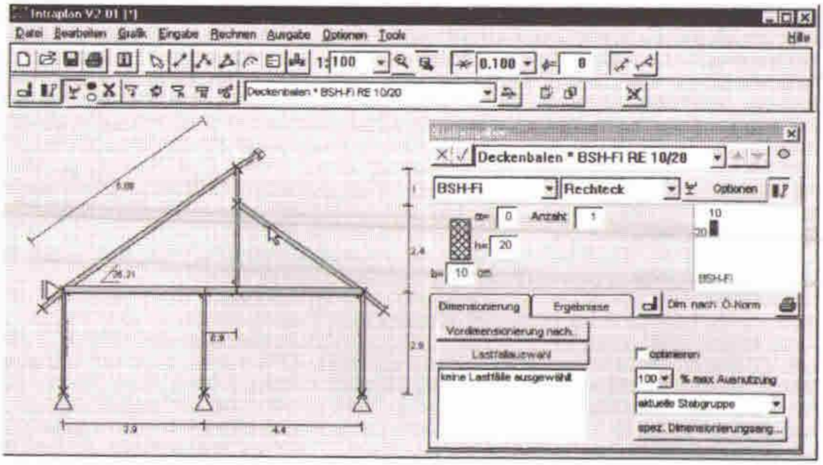
Joachim Hoffmann is a student of Physics at the TU Graz, Austria, who is increasingly working as an independent consultant and distributor for DyalogAPL, Causeway and J.

Appendix A: A short comic strip describing a typical structural design of a wooden house

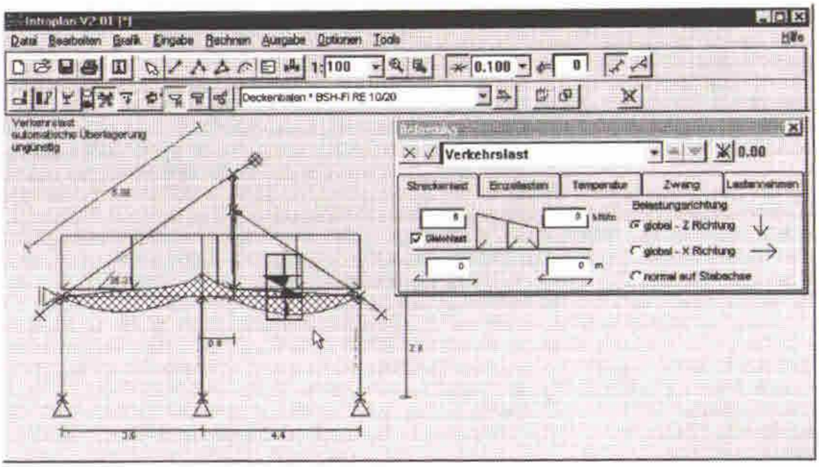
Input of the system with measurements and support definitions:


Intraplan V2.01 [*]
Datei Bearbeiten Grafik Eingabe Rechnen Ausgabe Optionen Tools Hilfe
Deckenbalen * BSH-Fi RE 10/20
Stablagerung
MN/m pro m Stablänge
Knotenlagerung
Einheiten
Darst

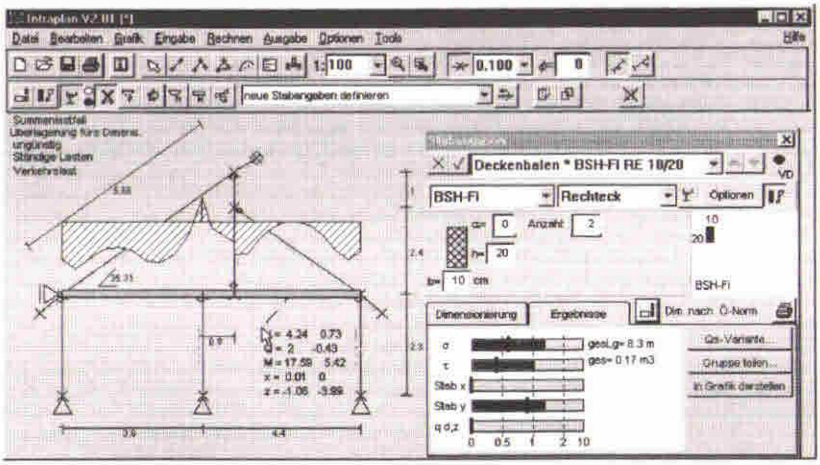
The assignment of member specifications: To each member a cross-section is assigned.


Intraplan V2.01 [*]
Datei Bearbeiten Grafik Eingabe Rechnen Ausgabe Optionen Tools Hilfe
Deckenbalen * BSH-Fi RE 10/20
BSH-Fi
Rechteck
Optionen
Anzahl
Dimensionierung
Ergebnisse
Dim. nach Ö-Norm
Vordimensionierung nach...
Lastfallauswahl
keine Lastfälle ausgewählt
optimieren
% max Ausnutzung
aktuelle Stabgruppe
spez. Dimensionierungsang...

The definition of loads: Traffic loads (e.g. the people in the building), snow loads, and wind loads:



The optimised load bearing structure:



Appendix B: German-English Vocabulary

German	English
Ablauf	process, mechanical operation or action
Abschnitt	section, portion, part
Abteilung für Holzbau	department for wooden construction
allgemein berandeter Querschnitt	general outlined cross-section
Annahme	assumption
Anzahl	number
Auflagerreaktion	support force or reaction
Auflagerverschiebung	buckling settlement
Auflagerwiderstand, -reaktion	reaction
aufskizzieren	sketch out
aufwendig	costly, expensive in labour
aus Gewohnheit	from (sheer) habit
Ausgangssituation	starting position, motivation
Ausnutzungsgrad	load factor
Ausschrift	display, presentation
Automatisierung	automation
Bauteil	component, device, construction element
Bauwesen	building trade
Beanspruchung	load, stress, strain
Beanspruchung auf Biegung	work done on bending
Belastung	loading, load application, stress
bemessen	dimension
Bemessung	dimensioning
Berechnung	computation, calculation
Berechnungsablauf	calculation method or operation
Berechnungsergebnisse	calculation results
besondere Aufmerksamkeit schenken	to pay attention to
Dimensionierung	design, dimensioning
Dipl.-Ing.	chartered engineer (GB), professional engineer
Dissertation	PhD thesis, MSc thesis (Master of Science)
ebener Rahmen	(flat) frame
ebener Verformungszustand	plane strain
ebenes Rahmentragwerk	plane frame structure
ebenes Tragwerk	plane structure
einbinden	integrate, include
Einzellastfall, Einzellast	point load
Ergebnisausschrift	results table
ermitteln, bestimmen (nachweisen)	determine (detect)
Ermittlung	determination
Ersatzstab	transformation member
Ersatzstabverfahren	equivalent beam method
etw. immer wieder tun	to keep on doing something
Fachwerkträger	truss (-ed girder), framework truss
Federkraft	elastic support, elastic force

Fichte	spruce, whitewood
freihandkonstruieren	freehand constructing
gekrümmt	bent
Gelenk	hinge
Gelenkknoten	hinged point, hinged connection
Generierhilfen	generating aids
Gewohnheiten	habit, to have a habit of doing something
halbvoll	half-filled
Holzbau	wooden construction
Holzbauteil	timber (structural) elements
immer wieder	again and again
immer wiederkehrend	recurrent
Ingenieursalltag	the everyday life of an engineer
Interessensgemeinschaft	interest group, syndicate
Knicklängen	buckling length
Koordinatenlisten	co-ordinate tables
Kranlast	crane load
Kreisbogen	circular arc
Lager, Auflager	support
Lagerbedingung, Auf~	support condition
Lagerung	support
Längeneinheit	unit of length
Längsträger	longitudinal girder
Lastananhmen	(assumed) design load
Lastaufbringung	load application, application of load
Lastband	field of load, load band
Lastfall	load(ing) case
Lastfallüberlagerung	load case superposition
Lastlinie	load line
Lastverteilung	distribution of load
LKW-Lasten	truck load
min-max Überlageung	min-max superposition
Oberkante (eines Trägers)	top surface
Parabelbogen	parabolic arc or curve
praktisch	practical, useful, in practice
Querschnitt = QS	cross-section, cross-section profile
Querschnittsangaben	cross-section properties, specifications
Querschnittsannahme	cross-section estimation (assumption)
Querschnittsfaktor	form or shape factor, cross-section factor
Querschnittsfläche	cross-sectional area
Querschnittskonturen	contours (outlines) of cross-section
Querschnittswert	cross-section value
Querspannung	transverse stress
Quersteifigkeit	transverse rigidity
Rahmenprogramm	frame program
räumliches Tragwerk	space frame structure, three dimensional frame
Referenzmaterial	reference material
Schätzung	estimation

Schere	scissors
Schneidefunktion	intersection function
Schnittkraft	internal force
Schnittkraftbilder	internal force diagram
Spannung	stress, tension
Stab	member, bar, element
Stabangaben	member specifications
ständige Lasten	permanent loads
Statiker	engineer engaged in structural calculations
Statikverarbeitungsprogramm	structure processing program
statische Problemstellung	structural way of looking at a problem
Streckenlast, Linienlast	linear load
Summenlastfall	cumulative load case
Systemfindung	developing a system design
Systemskizze	sketching
Träger	beam, girder
Trägerhöhen, veränderliche	variable girder depth
Trägersystem	girder construction, structural system
Traglast	limit load
Tragstruktur	supporting structure
Tragwerk	load bearing structure, supporting framework
Tragwerksplanung	design of frameworks or structures
Überlagerung	superposition, (awkward and) involved
Überlagerungsangaben	superposition details (specifications)
umständlich	laborious
ungünstige Überlagerung	worst case superposition
Unterkante	lower surface
Verarbeitung (DV)	processing
Verformung	deformation, deflection
Verkehrslastfall	traffic load
Vermaßung	measurements
verwalten	manage, conduct, maintain
Verwaltung	administration, organisation
Vordimensionierung	preliminary dimensioning
Vorgang, Prozess	process, action
Vorspannung	preload
Weiterverwendung	further processing or manipulation, subsequent treatment
Windbeanspruchung	wind stress
Windbelastung	wind load

The Logical Piano of W. S. Jevons

by Keith Smillie (smillie@cs.ualberta.ca)

[M]athematicians are well aware that their science, however much it may advance, always requires a corresponding development of material symbols for relieving the memory and guiding the thoughts.

W. S. Jevons

A mechanical device developed in the 1860s by the British political economist and logician W. S. Jevons for solving logical problems is described together with its simulation in J.

Introduction

William Stanley Jevons was born in 1835 and educated at University College, London having interrupted his studies to spend five years at the British Mint in Sydney, Australia. He then went to Owens College, now the University of Manchester, as a tutor but was soon appointed "professor of logic and mental and moral philosophy" and also "professor of political economy". In 1876 he returned to University College as professor of political economy, a position he held for only four years when he resigned because of poor health. He died two years later in 1882.

As an economist Jevons is remembered for his pioneer work in the concept of marginal utility and as the originator of the theory that trade cycles can be correlated with sunspot activity. As a logician he was one of the first persons to recognize the importance of the work of George Boole. His papers on logic were published in 1890 as *Pure Logic and Other Minor Works* which was reprinted in 1971.

In order to overcome what he considered an awkwardness in Boole's notation Jevons developed his "method of direct inference" which was a primitive type of truth table and similar to the logic diagrams of his contemporary John Venn. In this method Jevons represented up to four terms by the upper-case letters A, B, C and D, and their negations by the corresponding lower-case letters a, b, c and d. He would then write down all possible combinations of the letters and their negations, e.g., 16 combinations for four terms, 8 for three terms, etc., and then cross out all those terms inconsistent with the given hypotheses. From the terms remaining he could deduce what followed logically from the hypotheses.

At first Jevons wrote down for each problem a list of of all the possible combinations. He then devised a reusable "logical slate" with the combinations permanently inscribed so that the inconsistent combinations could be crossed out with chalk. Next he devised a "logical abacus" with the terms written on individual strips of wood which were placed on a rack. By means of a system of pegs on each side of the rack and a ruler, strips corresponding to inconsistent combinations of terms could be removed. It was Jevons's intention that this device be used in the classroom for demonstrating the solution of logical problems including Aristotelian syllogisms.

It was a relatively short step to completely mechanize this last device to produce what has come to be called the "logical piano". Jevons had this device built for him by a clockmaker in 1869, and demonstrated it the following year at a meeting of the Royal Society of London where he gave a paper entitled "On the mechanical performance of logical inference". The paper was published the same year in the Society's Philosophical Transactions and was included in the collected works referred to above. An excellent discussion of Jevons's logical work is given in Gardner (1982).

Jevons's logical piano is now in the Museum of the History of Science in Oxford. A few years ago it was on display on a windowsill together with a few gears from one of Babbage's machines.

In this paper we shall apply Jevons's method to the solution of a well-known syllogism with the steps presented in the way that they would be performed with the logical piano. We shall then discuss a simulation in J and illustrate its use with this example and with an example taken from the works of Lewis Carroll.

A Simple Example

Let us consider the syllogism Barbara which may be stated as follows:

Man is mortal.
Socrates is a man.
Therefore, Socrates is mortal.

If we let A = Socrates, B = man, and C = mortal so that a = not-Socrates, etc., then the premises may be stated as

A *is* B
B *is* C .

Now let us write down all possible combinations of the three terms and their negations as:

```
A A A A a a a a
B B b b B B b b
C c C c C c C c
```

For the first premise, A is B, the combinations involving a, the negation of A, are not relevent and may be set aside. Of the four remaining combinations,

```
A A A A
B B b b
C c C c
```

the third and fourth are inconsistent with the premise and may be discarded. If we combine the two remaining combinations with the four that were set aside, we have the following combinations consistent with the first premise:

```
A A     a a a a
B B     B B b b
C c     C c C c
```

Now proceed in a similar manner with the second premise, B is C. We first set aside the combinations involving b so that we have

```
A A     a a
B B     B B
C c     C c
```

then remove the second and fourth of these which are inconsistent with the second premise, and finally include those that were just set aside so that we have the following combinations consistent with both premises:

```
A       a   a a
B       B   b b
C       C   C c
```

We may conclude from the first combination that the conclusion of the premise, i. e., Socrates is mortal, is valid.

The consistent combinations may be written as

```
ABC + aBC + abC + abc
```

which may be simplified to

```
ABC + a(C + bc)
```

Unfortunately, Jevons's method of direct inference did not provide for this simplification.

The Logical Piano

The logical piano resembled a miniature upright piano about three feet high with a keyboard at the bottom above which was a display with four rows and sixteen columns giving all of the possible combinations of four terms and their negations. As the keys were pressed to enter each hypothesis, invalid combinations of terms would be removed mechanically from the display. When all of the hypotheses had been entered, only those combinations of terms consistent with all hypotheses would remain displayed.

The keyboard had twenty-one keys which may be represented as follows:

```
    R+dDcCbBaA=AaBbCcDd+.
```

For typographic convenience we have used R instead of FINIS which was used by Jevons, + instead of·|·, = instead of COPULA, and . instead of FULL STOP. The key . represents the end of a premise or hypothesis. Pressing the R key would reset the piano to display all sixteen combinations of four terms. The first premise in the example given in the previous section would be entered as `A=B.` and would result in the display of the six terms, two involving A and all four involving a, consistent with the premise. Similarly the second premise would be entered as `B=C.` Subjects or predicates involving two or more terms could be entered as, for example, `AB=C.` or `A+B=C.` depending on whether logical 'and' or 'or', respectively, was implied.

Figure 1 shows the Windows form representing the J simulation of the logical piano. The box outlined below the keyboard has been added to the simulation to record the hypotheses as they are entered. Also added to the simulation is the option of displaying those combinations available for two, three and four terms. For example, the eight combinations for three terms may be displayed by selecting the button for three terms and then pressing the R key. Figure 2 shows the logical piano after the premises for the example of the last section have been entered.

The script file for the simulation is available by anonymous ftp at *ftp://ftp.cs.ualberta.ca* in the file *pub/smillie/piano.js.*

A Second Example

The logical piano may be used to solve some of the problems given in Lewis Carroll's *Symbolic Logic*. As an example consider the following problem where the object is to determine the consequences of the three hypotheses:

1. No one takes in *The Times* unless he is well-educated;

2. No hedge-hogs can read;

3. Those who cannot read are not well-educated.

   Univ.='creatures'; A=able to read; B=hedge-hogs; C=taking in *The Times*; D=well-educated. [We have used the upper-case letters A, B, C and D to be consistent with the notation introduced above whereas Carroll used the corresponding lower-case letters.]

The three conditions may be represented in the notation of Jevons's logical piano as `C=D.`, `B=a.` and `a=d.` Figure 3 shows the use of the piano simulation for the solution of this problem. We may conclude, for example, from the combination `aBcd`, the only combination involving `B`, that hedge-hogs are not able to read, do not take in *The Times* and are not well-educated.

Postscript

Jevons considered building a machine that could accommodate ten terms but abandoned the project when he realized that it would take up one entire wall in his study. In 1881 Alan Marquand, tutor of logic at Princeton University, built an improved and smaller version of Jevons's machine, and in the following year with the help of a friend in the Department of Mathematics he built a more elaborate version. Marquand also proposed a third electromechanical version but because of difficulties with the new electrical technology did not get beyond a prototype built from a hotel annunciator.

A few years ago two American academic economists wrote the mysteries *Murder at the Margin* (1978) and *The Fatal Equilibrium* (1985) with the purpose of teaching some of the principles of economics and at the same time entertaining the reader. The authors wrote under the pseudonym "Marshall Jevons", where the surname comes, of course, from W. S. Jevons and the given name from the British economist Alfred Marshall (1842-1924). These two books may be recommended not only to readers of mysteries but also to those who believe that some lightheartedness in learning is a good thing.

References

Aspray, William (ed.), 1990. *Computing Before Computers.* Iowa State University Press, Ames, Iowa.

Fisher, John (ed.), 1975. *The Magic of Lewis Carroll.* Penguin Books Limited, Harmondsworth, Middlesex, England.

Gardner, Martin, 1982. *Logic Machines and Diagrams.* Second Edition. The University of Chicago Press, Chicago.

Jevons, W. S., 1890. *Pure Logic and Other Minor Works.* (Reprinted by Burt Franklin, New York, 1971.)

Keith W. Smillie
Department of Computing Science
University of Alberta
Edmonton T6G 2H1, Canada

Figures

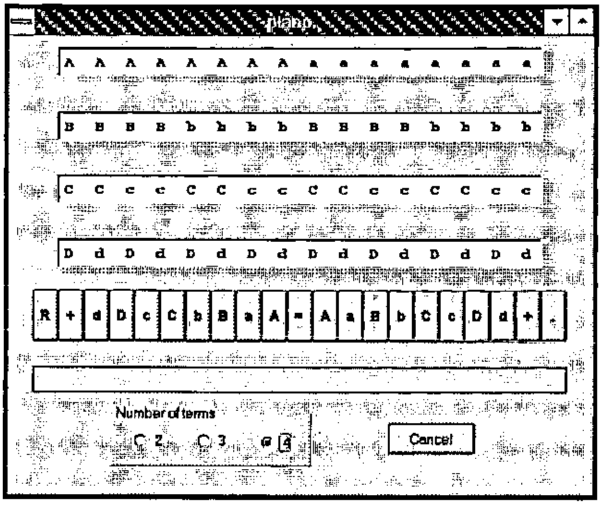
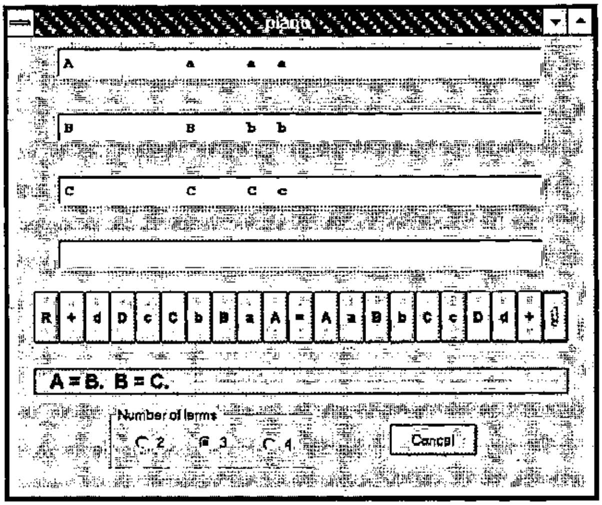

piano
A A A A A A A A a a a a a a a a
B B B B b b b b B B B B b b b b
C C c c C C c c C C c c C C c c
D d D d D d D d D d D d D d D d
R + d D c C b B a A = A a B b C c D d + .
Number of terms
2
3
4
Cancel

Figure 1. Simulation in J of the logical piano

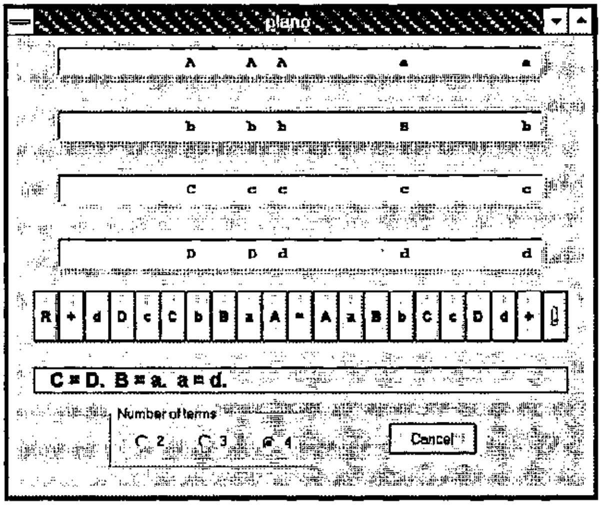


```
A               a     a a
B               B     b b
C               C     C c
```

A = B. B = C.

Number of terms: 2 [✓ 3] 4

Figure 2. Solution of Aristotelan syllogism



```
                A     A A          a          a
                b     b b          B          b
                C     c c          c          c
                D     D d          d          d
```

C = D. B = a. a = d.

Number of terms: 2 3 [✓ 4]

Figure 3. Solution of Lewis Carroll problem

APL97 Conference Survey Results

analysed by Dick Holt (dholt@capaccess.org)
(This article has already appeared in several local APL newsletters)

This article summarizes the APL97 attender feedback survey. It compares results, where possible, with similar prior surveys. It describes attenders' satisfaction with various features of APL97. The purpose of this article is to help the APL/J community to improve future conferences.

Briefly, attenders rated APL97 as a huge success. 10 of 14 Conference features rated higher than in two comparable prior surveys, most of them notably so. The 4 exceptions (below), may appear to be comparable to prior surveys, but for varying reasons, are not.

(1) "Proceedings" rated slightly lower at APL97 than in the 1994 survey (Proceedings were unavailable at APL96). Some APL97 attenders noted that Proceedings on a CD ROM and weren't easily readable on-site (but only 30% looked at the CD ROM — see Q17). Even so, the CD ROM helped APL97 to reduce lead times and to keep costs low. The APL97 Proceedings rating reflects its format, not its content. As a result, it's not comparable to prior survey data.

(2) In contrast, APL97 attenders rated the Software Library highly compared to prior surveys. Again, this reflects the CD ROM format rather than content.

(3 & 4) "Cost of conference" and "Other costs" aren't comparable to prior surveys. We changed the APL97 questionnaire to distinguish, for the first time, between "Cost of conference" (e.g. registration and banquet), and "Other costs" (e.g. travel and lost earnings). "Cost of conference" at APL97 rated in 1st place, clearly showing APL97's success in reducing those costs over which its organizers had control. Yet, "other Costs" rated low. In statistical jargon, APL97 data for "Cost" and "Other Costs" are confounded if compared with prior survey data.

Now to the Survey Questions and Answers

Q1 asked attenders to rate APL97 features on a scale of 1 to 5 (most-to-least satisfied). Average ratings, listed in order of decreasing satisfaction, are shown below:

Table 1. Satisfaction with APL97 Conference Features

cost of conference	1.10	x	social/vacation	1.77	*
location	1.29	*	papers	1.81	*
website	1.32	n	proceedings	1.84	x
program handbook	1.36	n	other costs	1.89	x
archives, etc.	1.43	n	dinner cruise	2.03	*
tutorials	1.58	*	business opportunities	2.06	*
software library	1.59	x	job information	2.19	*
accommodation	1.68	*	vendor hospitality	2.34	n
workshops	1.72	*	posters, displays	2.58	*
vendor forums	1.74	*	others? — only 1 response		

* = ratings higher than 94/96 surveys
n = not asked in 94/96 surveys
x = not comparable to 94/96 surveys

Small differences in ratings (+/- 0.2) probably aren't significant.

See References 1 and 2 for 94/96 survey data.

Q2: What was the most useful feature, and why? Multiple responses are combined below:

- Arrangements: Very good arrangements; good programs; program was well thought out
- Workshops: learned about very pertinent stuff; got to try things; best way to learn is to try; participants received real hands-on experience and training in advanced features.
- Tutorials: practical how to do; excellent content; stream of J tutorials; subjects were mostly practicable; APL info.
- Workshop and vendor forums
- Convenient (for us in Toronto). Very knowledgeable crowd.
- Vendor forums: clear understanding of current market; got the latest developments
- Meeting people
- CD rom proceedings — may make software accessible, pretty current

Q3: What was the least useful feature, and why?

- Vendor forums too far from presentations (multiple comments)
- Workshops — great idea but poor presentation. Overhead info too dead, too small and too dim to be read or used.
- Workshops — topic selection poor for me

- CD rom proceedings — can't read papers at conference or on trolley; Missing proceedings (see Q17); hosp suites instead of open is best.

Q4 and Q5 concern Conference attendance. Overall, 79% of respondents had attended a prior SIGAPL conference, and 84% said that they would attend a future one (90% and 92% at APL96).

When asked why they might attend a future SIGAPL Conference, respondents said:

"I use APL and J professionally; Fun and fruitful; It's my business; It has become a habit; Interested in APL employment and meeting other APLers; Greater APL involvement; Useful info; Worth attending; Business; Helps to keep abreast of developments; For APL related information, products, contacts; Meeting people; Stimulating to get new ideas."

Q6: Are you a member of SIGAPL? Y=70% (APL96: Y=90%)

Q7: Years of experience with APL? With J? A simple answer: an average of 13.6 years with APL, and 2.7 years with J. Prior surveys showed a trend of increasing years of APL experience (15.6 in 94, 16.9 in 96). That trend is lightly reversed at APL97.

A more interesting answer: about 28% of respondents are experienced in both APL and J. Their joint years of APL and J experience are shown in the 2x12 table below:

```
APL yrs:  6  20   8  15  14  21  25   8  13  20  20  25
  J yrs:  2   6   1   5   1   5   3   2   1   1   6   1
```

Similar cross-tabulations, mainly of interest to vendors, may be constructed using the APL97SRV.ZIP file mentioned below.

Q8: Your State/Province/Country? This question is key to the analytic validity of this article. These analyses are valid only if the questionnaire respondents are a statistically representative of the population of all APL97 attenders. A 10,000 draw bootstrap (Monte Carlo) shows that the survey sample is indeed representative of APL97 attenders.

	# of APL97 Attenders from:	# of Survey Responses from:	Bootstrap mean	Bootstrap 2 sigma
Canada	96	11	16.5	5.8
USA	82	18	14.1	5.6
Europe	61	12	10.5	5.0
Other	11	2	1.9	2.4
Total	250	43		

See Reference 2 for details of the bootstrap technique. The US is somewhat over-represented, and Canada is somewhat under-represented in the sample, however both are within 2 sigma limits of their bootstrap means. "Europe" and "Other" are easily on the mark. Attenders came from 4 Canadian Provinces (mostly Ontario), 20 US states, 10 European nations (including Russia), and from Japan, Saudi Arabia, and South Africa. The survey response rate was 17%, well below the 27% response rate for the APL96 survey. Why? Because APL97 survey forms were available only adventitiously, rather than being inserted in Conference Bags at registration.

Q9: "Your occupation?" showed a wide array of responses. Most were in the area of software development (32%) and education (16%), both very close to APL96, followed by actuary (7%) and scientist (7%). Others include student, engineer, financial analyst, webmaster, consultant, business, and others.

Q10: "In what applications have you used APL/J?" Applications mentioned most often were:

banking/finance	.39	insurance/actuary	.09
education	.30	engineering	.07
business/manufacturing	.12		

Many respondents named multiple applications including advertising, medicine, oil, law, elections, beverage, forestry, travel, biology, nutrition, and others.

Q11. Are you actively developing code in APL, J, or other Array languages?

APL only : 56%	J only : 7%
Both APL and J : 9%	Other array also: 14%

Q12: Where did you first hear about APL97?, and Q13: What was your major source of ongoing information about APL97?

	mailing	email	c.l.a.	Vector	QQ	Nouvelles	Other
Q12:	14%	26%	19%	2%	5%	0%	23%
Q13:	16%	53%	21%	7%	7%	5%	14%

APL97 publicity reached far and wide. Web sites were most often cited as "other." Electronic publicity was the most wide reaching.

Q14 and Q15 concerned Vendors. 56% of survey respondents visited vendor Hospitality Suites, and of those who visited, 86% felt that they provided a useful contribution to APL97.

Q16 asked for comments on Exhibitors/Vendors. Opinions were mixed, and physical location was apparently an issue:

On the plus side:

> "Hospitality was one of the main reasons for attending, finding out about new products and watching them demonstrated; Suites harder to get to but receive more attention there; More emphasis on the seriously interested; Hospitality is better; Hospitality is better more focused."

On the minus side:

> "I prefer ... a single exhibitor area. This was too scattered; Exhibits were close, hospitality was remote; Prefer central open exhibit area, really *near* lecture rooms; Too far from meeting room; Hospitality suites too formal ... a barrier to go there; Wish vendor sites near coffee between papers; Hospitality suites should have been in same building."

One respondent noted that the survey didn't distinguish among vendors, and that some vendors had better hospitality suites than others. Almost all one-word responses to Q16 favoured the "open-concept" over the hospitality suite approach for Vendors' exhibits.

Q17: CD ROM: Only 30% of respondents looked at the CD ROM, and 60% of those who did found it useful. But this isn't the full story. Many noted that they couldn't easily read the CD at the Conference.

Below is an example of suggestions for future CD ROMS. See also Q3 (least useful feature).

> "(1) The CD was a good idea but it went too far: Please distribute attendance list ON PAPER at start of Conf (later a more updated version on www is fine too). (2) I do like to have the proceedings ON PAPER. If not, my boss will not consider any contribution I make to the Conf ... as valuable and will not pay for my trip. (3) I do like to have the proceedings on paper to read on the first night of the conf. (4) I appreciate a low cost conf, but am willing to pay more for proceedings on paper."

You may order the APL97 CD ROM from www.torontoapl.org for $C20, postage paid world-wide, via cheque, money order, Visa, Master, or Amex. Toronto SIGAPL will make the CD ROM electronically available and updated at, or linked to, its web page (www.torontoapl.org).

Q18: Was the Program Handbook useful? Y = 74%

Program handbook comments:

> "Add good map of the area ... is needed; Pg 18-20: begin-end times, like in the flyer, easier to read than the begin time only; Set start times on print-out more carefully; Suggest adding start and end times; Keep schedule in center — easy to open and find; Put day, time, and location beside each description to change more bigger one size; Perhaps include a floor map of room locations; It was exactly up to its task."

Q19a: Want Annual APL/J conference in Toronto? Y=28% N=28% B=44% Q19b: Want a Bi-annual conference? Y=49% N=19% B=37% Note: some respondents may have interpreted the word "bi-annual" (twice/yr) in **Q19b** to mean "biennial" (every 2 years).

Q20a: Are you aware of the APL Skills Database? Y=62%

Q20b: Other services wanted? There were a lot of blanks in response to Q20b. Items mentioned were:

> "Part-time employment; more web stuff; Résumé marketing service?; internet conference; (1) debugging tools, (2) Ecklers (Paul Davidson) coding protocols, (3) version control software; and workshop, — contest"

Q21: Topics for future conferences?: Verbatim responses were:

> "Performance consideration in real-life software developed in APL/J; Application in scientific & mathematical problems; OLE controls (i.e. getting them to work in APL); General tips on programming style, efficiency, etc; Language and interface enhancements in APL; working products; How J/APL is useful; Making APL/J useful to end users (as opposed to developers); Success stories using APL/J as well as unsuccess stories; Vendors cooperation in research for improving APL."

Q22: Other comments or Suggestions? In the respondents' own words:

> "Keep up the good work; Thank you Committee! Ryerson's facilities, especially classroom were much superior to what was available in past at U.T.; It would be great to have a participants list in the registration kit at the beginning conference. Always is it good service (many other Conferences use it and I enjoy it); Encourage presenters to have a script beforehand. Some of the

projections via overhead was unreadable. Need to schedule beforehand. Great conference. Thank you; Allow a 1.25 — 1.5 hours for lunch. Lunch is too short (several comments); Ran out of coffee/refreshments during break; Great Conference, all the organizers deserve a lot of credit; More interesting content than any recent conf; Thank you for a great conference; Thanks for your effort in sponsoring APL97. It was certainly one of the best ever; Overhead projections not readable in 90% of the workshops, Sit at the back of the room to test for yourself; I compliment Toronto SIG on a very well done conference, including location and accommodations, and program pamphlet; Advanced applications are nice, but for the average programmer, the problem is: 1) ease of reading code, 2) debugging it, 3) with multiple programmers, how to keep their work organized; Some workshops were less useful as all we did was typing ... creating images in J which left us some assignment. We were too busy typing to follow the message/contents of the workshop. Suggestion: reduce the number of terminals and operators; Good Conference; An excellent conference. The program committee should be commended for a job well done."

File APL97SRV.ZIP containing all analysis fns and APL97 survey data, is downloadable free from the BBS\APL at 703-528-7617, in File Area FREE. APL97SRV.ZIP has also been sent to Toronto SIGAPL and Waterloo for electronic distribution.

References

[1] Dick Holt, *"APL96 Conference Survey Results"*, APL Quote Quad V27, No. 1, September 1996, pp.36-38

[2] Dick Holt, *"1994 SIGAPL Survey Results"*, APL Quote Quad V25, No. 3, March 1995, pp.9-15

Dick Holt can be reached at dholt@CapAccess.Org, fax 703-528-7617, or mail at 3802 N. Richmond St. Arlington VA 22207 USA

GENERAL ARTICLES

This section of Vector is intended for general readers who have a working knowledge of APL or J. Authors are encouraged to submit articles which illustrate effective APL applications.

How to Win Programming Language Battles...and Maybe the In-House Applications Development War

by Donald B. Pittenger (dbpitt@msn.com)

A good start to winning the next war is to have lost the last war. APL lost its war. But it can be part of a team that can win the next one. This article presents one prospective strategy and related set of tactics. It might well infuriate you, but it will be for a good cause if it inspires you to come up with a better plan.

I suppose I should explain the meaning of the first sentence for readers not familiar with military history. The concept is that winners tend to rest on their laurels — after all, they won, so they must have done things right. Losers often realize they did things wrong, and study their defeat in order to correct defects and do better next time. (This does not always happen if correcting the key problems is beyond the capability of the loser; a Belgium without allies is always almost certain to be defeated by a Germany. Other defeated nations tend not to self-examine, but place blame elsewhere: 'We are betrayed!')

The first step to winning the next programming language battles is to admit that the previous war was lost. So far as I can tell, almost all APLers concede that APL will not rule the programming world, as they had hoped in the late 1960s and early 1970s. Some have given up completely. Others have been scurrying to survive, popping out useful ideas and sales points — but the impact has been scattered and small.

I will start with a limited outline of the defeat. Then I present my interested bystander's perspective on the current situation, followed by a plan for the next campaign.

The Defeat

What is history cannot be undone. Even if the betrayal theorists are correct that IBM (consciously or unconsciously) did APL in by promoting PL/I in the early 1970s, that does not matter; besides, today PL/I seems to be more out of the picture than APL! Aside from the PL/I theory, what went wrong?

I am sure the data processing management view that APL systems were unpredictable mainframe memory-hogs was based on experience. Hostile management does not guarantee defeat, but it makes life pretty hard. I think this is what started APL's slide beginning in the mid-'70s. DP management was also hostile to personal computers, but the sheer sense of the PC concept along with the killer-application spreadsheet and its champions in other realms of corporate management more than neutralised DP's position.

When PCs did come along, APL was past its usage peak, and early-'80s PCs were too small and slow to make effective use of APL. It was not until the late 1980s and the advent of 32-bit CPUs and simple memory mapping that APL had a chance to shine; but, by then, the damage had been done.

Language-specific problems are well-known, even though many APLers do not consider them to be problems. These include: (1) the character set — elegant, but until recently not easy to integrate into the computing environment; (2) the fact that APL is interpreted rather than compiled — unless the computer itself is lightning-fast, commercial 'shrink-wrap' applications are to be avoided in APL; (3) the APL workspace — a convenience in interactive computing, but its operating system-like qualities have made APL hard to integrate with host OS's, particularly in application development. There are other intrinsic problems, but the above were more than enough keep APL sliding over the past two decades.

The Present Situation

Where do matters stand today? Computer technology has continued behaving according to Moore's Law: speed increases while cost of equipment (of a given performance level) decreases. Many desktop computers have the speed and memory to overcome slow interpreter speed perception in many application settings. Regarding operating systems, APL is often packaged as simply another library or library set (DLLs in Windows) and can communicate with other applications and code libraries. The symbols remain controversial and a potential problem so far as keyboards are concerned for the programmer. But symbols and keyboard drop out of the picture in applications — which was true even in mainframe days.

An important fact is that a number of APL-offshoot languages have sprung up since the mid-1980s. These include Nial, J, A, A+, K, and the SAS IML language. Mathematica claims more remote kinship. Apart from A and perhaps A+ (I haven't seen A+ code), these languages do not use the APL character set. J has eliminated the workspace and strives to link with all manner of other applications on the host computer or even over the Internet. A+ and K are number-cruncher languages tailored to the needs of Wall Street firms; this too is important, as we shall see.

And, of course, it is well known within the array-language community that programming (if not execution) speeds are far greater than for conventional computer languages.

Given the above as well as other factors not covered for lack of space, many prerequisites for a successful renewed struggle are in place or nearly so. What is missing is a plan for taking advantage of the situation. We turn to that next.

The Plan: Introduction

Here we bring in marketing and public relations, trying to effectively blend them into the mix of technology and office politics discussed thus far. And please don't let your dislike of 'suits' prejudice your reaction to what I am about to discuss. Marketing does not have to be the slick stuff that many APLers disliked about STSC/Manugistics in the era 1985-95. Properly done marketing and PR, coupled with acceptably good technology, can take you far. You cannot deny the success of Microsoft as a case in point.

The plan is presented in three parts. First, strategy is outlined; this is likely to be the most controversial point. Then some tactical steps are proposed. Finally, there are a few cautionary notes.

One caution can be noted at the outset, namely: in war, plans seldom survive contact with the enemy (read von Clausewitz). My notion is that the plan should be simple, but not so tightly focused that unanticipated successes cannot be exploited due to its mindless pursuit ('exploit success, not reinforce failure' is useful military maxim). And remember that even simple plans are very difficult to execute (I think this is from Napoleon).

Strategy

APL is mostly used for: (1) in-house applications; (2) preparation of data for access or sale to outsiders — i.e., the LegiSlate information system, the demographic data I sell; and (3) *ad hoc* interactive computing by engineers of scientists. 'Shrink-wrap' APL-based products hardly exist. *In-house application development is clearly the strategic target* because it is a huge field and it plays to APL's strengths (efficient development, new abilities to link to other software) and minimises its weaknesses (shrink-wrapping and slow execution do not seem to mix well). If there is success in the in-house arena, the data product and *ad hoc* areas should benefit as side effects.

Now for the controversial part. *APL's image* (such as it exists today) *largely stinks.* I see no easy way to turn that image around by focusing on APL in isolation.

This means APL should hardly – if ever – be mentioned in our rehabilitation campaign.

Instead, we promote Array Languages as the next big thing in information systems productivity.

And, since Array Languages are more than just APL, we *APLers must be able and willing to write in J, K, Nial, or whatever* good new Array Language comes along. It can become a marketable skill area.

All Array Languages are small players, and uniting them for public relations purposes is likely to yield more short-run benefit to all than promoting each in isolation. If the Array Language concept becomes a hit, then the alternative languages can fight over a much larger market than before.

Let me summarize the strategy: Use the concept of rapid-development Array Languages as the next step in information processing. There was OOP for interfaces and data bases, now there's Java for the Internet, *next come Array Languages for core, mission-critical number crunching.*

Themes

The most important thing is to show that existing Array Languages are already being used successfully for core, mission-critical number crunching.

Where does this happen? Wall Street. Billions of dollars are at risk in lightning-paced trading situations. Very high stakes. *And what kind of computer language do key Wall Street firms turn to for this crucial task? – Array Languages, of course!*

To me, the above is a powerful sales point.

And it can be reinforced by other, similar points: *An Array Language was a key item in developing anti-missile defences – could your company use that kind of performance?*

I hope you get the idea — find the most impressive examples and stress the Array Language aspect. The actual Array Language used becomes a 'for instance'.

I also believe that practical appeals are stronger than theoretical appeals for the target audience. Yes, everyone can be swayed by emotion, and many can be swayed by ideas. But the key is to take a rational appeal and make it emotionally appealing. Thus, someone who buys into the concept of the Array Language solution can use the rational part of the argument as justification. The examples above are intended to convey the idea that Array Languages offer powerful solutions in settings where failure might be disastrous. The reader of the messages thinks his own work, even if on a smaller scale, is pretty important too; so he has reason to be motivated to learn more about the Array Language solution.

How is the message to be spread?

Tactics

Now we are at the hard part: execution. This is a black art that even marketing and public relations professionals do not fully understand, mostly because causes and effects are hard to trace and measure. (As an aside, the only really measurable advertising is direct-response, where the customer inquiry or purchase can be traced to a specific mailer, print ad, or television appeal — the contact address or toll-free phone extension varies from advertisement to advertisement and the effect of each ad is thereby measured.) Since we are doing public relations, we must fall back on industry folklore and published comments by successful practitioners.

There is time to think about this before taking action. Ordinarily, my instinct would be to start immediately, but there is the small problem that the computer world is abuzz with the Internet and the Java language. I think it would be a waste of resources be go full-blast with an Array Language campaign while everyone is still focused on Java. This does not mean that seeds cannot be planted; it means that the Big Push will fail if it is launched before Java becomes commonplace and the Next Big Thing needs to be found to occupy the attention of the computer elite.

What kinds of timelines exist for computer industry fads? My impression is that there is usually a period of gestation or build-up. This can take a decade or even more. For this reason, seed-planting should begin soon. The Java explosion (which began in 1995) is an exception. C had been around for several years before the rush to 'software portability' became the craze in the mid-1980s. C++ was little known until object-oriented programming exploded in the collective programming mindspace around 1990. The practice of OOP was launched at the famous Xerox PARC in the late 1970s (though the ideas were manifested in Simula in the late '60s), and *Byte* magazine had a cover story on the Smalltalk language in the early '80s. Graphical interfaces made the need for object-orientation concrete. C++ was the commercial winner thanks to the fact that C was, by that time, the prime language for shrink-wrap software, and moving from C to C++ was a smaller step than from C to something else. OOP is far from dead. The latest database concept is object-orientation (to supplement or complement the relational paradigm). And there have been both theoretical and practical efforts to make APL OOP-compliant. At any rate, OOP's gestation from concept to mania can be pegged at roughly a dozen years, if Smalltalk is taken as the start-point. The time from the invention of structured programming languages (Algol, around 1960), to the frenzy for top-down programming was about the same.

I stress fads because the fad is the expression of the attention and energy focused on a particular programming concept. And, unlike fads in clothing and automobile styling, programming fads tend to become honorably installed as part of mainstream practice once the hype wears off and the most extravagant claims are shown to have been false (for instance, pure-OOP, as exemplified by Smalltalk, has not taken over the world). Even APL has become a structured language. Top-down programming is the best approach for many tasks. Graphical interfaces are almost unthinkable without OOP elements. *And Java might legitimize interpreted languages.*

To return to my point: the groundwork for a surge in Array Languages has been going on since about 1990, when APL was about to become better integrated with other software on the host computer and when J and A were under development. This suggests that the technical phase is well underway, and that the promotional effort should begin soon, even before the Java hype starts to fade, with expectation of success around 2000-2003. We are approaching the window of opportunity.

One public relations strategy — advocated by Regis McKenna of Intel and Apple fame — is to capture the attention of 'industry influentials'. In the case of Apple Computer, this meant getting key venture capitalists to invest early on as well as gaining the favorable interest of Ben Rosen, at that time a respected industry observer and newsletter publisher (who later became involved with Compaq Computer from the VC side). McKenna contends that, thanks to their prestige and connections, opinions of influentials will seep down the chain to news media, corporate purchasers, and the general public. This concept worked for Apple, but it is not sure-fire, and might not even apply to our problem of publicising a family of programming languages rather than a particular branded product.

Nevertheless, programming language influentials exist and should be cultivated. Here are some fairly obvious examples to get your thinking launched. Consider Bill Gates. It is known that he was once intrigued with APL, so the idea of using an Array Language as part of the Visual Basic package or as an adjunct to the Access database product might not be rejected out of hand at the highest levels at Redmond, a few miles up the road from where these words are being written. I suppose Bill himself probably isn't a primary target — but if an Array Language buzz starts, some folks at Microsoft, including Bill, will surely be aware of it.

Besides Bill, there are the usual media suspects. Someone with a better sense of history than me can correct this, but my impression is that commercial computer magazines are better at following trends than creating them. But they too should be cultivated. And this cultivation will not be effective as a one-shot deal. I recall that, a few months after APL89, Ed Cherlin was able to get a nice amount of APL2 material into John Dvorak's column in *PC Magazine*. Trouble is, nothing else appeared, and the effort was wasted.

Here is a concept for dealing with the media. At some point, there should be a hard-hitting barrage to a number of columnists such as Dvorak about a new class of 'wonder languages', namely Array Languages. Leading up to that, word could be spread amongst programmers in general via magazines or newsletters (or Internet venues such as chat forums with computer magazine editors and columnists) about great new ways to get code out. An example might be Jeff Duntemann's magazine, *Visual Developer*. Its roots are as a magazine for Borland Turbo Pascal programmers, but it has evolved to focus on programming for Windows in Delphi and, yes, Visual Basic and Java. There are some good APL writers with experience of using J (or Dyalog APL/W) as DLL servers under VB, Delphi, and Visual C++. It seems to me that articles could be submitted to *Visual Developer* telling how really great, quickly-coded number processing routines can be attached to the various interface systems for use in one setting or another. If even one or two of these could reach print each year, that would be a reasonable start. And the effect would be multiplied if the same thing was happening in *Dr. Dobb's Journal* and similar magazines.

Besides programmers, we must start to capture the minds of corporate IS managers and systems analysts. This is getting away from my experience, so I will be sketchy and brief. Magazines and journals read by such people should be a target. The concepts relating to Wall Street mentioned above could have great impact here.

Yet another target is the computer guru or management consultant. Getting consultants to buy into the concept of (comparatively) rapidly-programmed number-processing applications using Array Languages as vital adjuncts to existing DP tools could be a large win for our side.

How is all this going to take place? The cheapest way is to form a volunteer committee to orchestrate the effort, to coordinate the solicitation of articles, inclusion of panels in non-APL professional meetings, etc. But volunteers can be hard to recruit, and volunteer committees tend to lack staying power — an important consideration for a multiyear effort. Nevertheless, the option proposed next might well be even more impractical; then the volunteer committee would be the only option.

In my opinion, a good way to do things is to form an *Array Language trade association,* either formally chartered or simply an informal group. An association could have people helping in the coordination, marketing, and PR efforts while on company time, thus taking some pressure off volunteers (who would still be needed for writing books and articles and for making presentations or appearing on panels).

Another benefit of a trade association is that it could pool some money to hire a public relations firm or consultant to deal with PR basics such as maintaining contact lists, brokering articles in trade journals, and keeping an eye open for opportunities to publicise the Array Language concept. The consultant also ought to be able offer advice to the association regarding PR matters in general.

My guess is that a low-level, initial effort might cost something like $10,000 per year. If industry sales increase, and the consultant can demonstrate media placements and other measurable results of his activities, then the budget and effort could be increased.

Cautionary Notes

Probably the most important consideration is to be able to deliver what is promised. Here, I am thinking mostly of the ability of Array Languages to link to host computer applications or to the Internet or whatever connections become fashionable in the next few years.

Another need is to better assure that programmers taking up Array Language coding have the resources (books, manuals, training opportunities) to get going reasonable quickly without much pain and anger. Bitter experiences can damage a product or cause quickly and thoroughly. The risk of such problems is high when programmers must drop old habits. This was the case with APL in the early days — programmers writing scalar APL code. (More recently, programmers from many backgrounds have had to face the pain of coping with the OOP paradigm. The need or desire to change fuelled the publication of many articles and books on the subject to ease the shift.) On the APL side, perhaps 'Gilman and Rose' could be resurrected; in its day, it was a Godsend. J is notoriously hard to learn, partly for intrinsic reasons, and partly because good introductory material has been lacking. The *User Manual* is good for learning the environmental parts of J, and the *Primer* should have appeared long before it did. At least one more introductory book by a different author is needed, simply because different perspectives work better for different people when learning new material. I am not familiar with learning resources for Nial and K.

Finally, I think it would be helpful if APL programmers were more open-minded regarding changing the language as well as their own work habits. I don't mean all APL programmers, of course. But each innovation (keyboard revisions, control structures, GUI environment) seems to lead to some angry letters to journal editors. As mentioned above, the Array Language concept can be the vehicle for exciting professional opportunities. The price of this might include abandoning component files, the workspace, and — yes — even the del editor to embrace evils such as control words.

Conclusion

We don't have to lose. Array Languages can be the next programming sensation. Perhaps I didn't get everything right in the way of diagnosis and prescription, but I hope it will stimulate folks in the APL community to seek new ways to spread the Array Language gospel to the outside world.

TECHNICAL SECTION

This section of VECTOR is aimed principally at those of our readers who already know APL. It will contain items to interest people with differing degrees of fluency in APL.

Contents

A PostScript Type 1 Font Downloader

by John Sullivan (john@yddraiggoch.demon.co.uk)

What is a Type 1 PostScript Font?

A Type 1 PostScript font is a special case of a PostScript language program, which tells a printer how to make each character of the font. The program uses characters from the lower half of the ASCII character set, but in order to save space on disks containing fonts, the part of the font that is encrypted into hexadecimal characters is compressed by a factor of 2.

Why do we need a Downloader?

A download program is required to undo this compression and send the font to the output device for use by other PostScript programs. The output device is usually a PostScript printer, but it could be a file if the output is not to be printed immediately.

In Microsoft Windows with Adobe Type Manager, the printer driver will sort out all the nuts and bolts of sending installed fonts to the printer for you. But what do you do if the font is not installed (there is a limit to the number of fonts you can install), or if you are not using Windows, or any other software which handles printing for you? When I started home computing I used an Atari ST, and this did not have any way of sending fonts to a PostScript printer, so, I had to find a way of downloading fonts myself. The information is not generally available, although the books published by Adobe Systems themselves[1] do contain some useful information.

Gallia est omnis divisa in partes tres

So said Julius Caesar[2], but he might equally well have been referring to a PostScript font! The middle part of the font, called the "eexec encryption" is surrounded by two parts in plain ASCII. Each of these parts starts with a two-byte delimiter and a four-byte length, and the whole font file ends with another two-byte delimiter. The task of the downloader is to split the file into the three parts, uncompress the middle one, and put it all together for sending to the printer.

The Downloader

The downloader is written for Dyalog APL but it should be easy to convert to any other APL (as mentioned above, it started life as APL.68000 for the Atari ST, so it has been converted once already). It takes the raw font file as its argument and returns the downloadable font. No I/O is performed in this function because that is for the programmer to sort out in the application. All I/O should be performed with no translation, which is why line 1 of then function looks odd (it should end up on the printer as `serverdict begin 0 exitserver`). This version assumes that the lengths of each part of the font file are stored in Intel format: my PostScript fonts were bought for DOS/Windows, and fonts for other platforms may have the lengths stored differently. If this is the case with your fonts, make the appropriate changes to variable `c`.

This function does not check that the input is a font file because it assumes that that has been done already. The index origin must be 0.

```
    ∇ z←fontload x;a;b;c
[1]   z←⎕AV[4],'|È┌ãÈ┌Çɪ├ρ┬È⊟ɪ⊞ρ0ρÈ⍤ɪ┤|È┌ãÈ┌',⎕AV[13]
[2]   z←z,x[6+⍳a←II x[c←5 4 3 2]]
[3]   z←z,'0123456789ABCDEF'[,⍉0 16⊤⎕AV⍳x[12+a+⍳b←II x[a+6+c]]]
[4]   z←z,x[18+a+b+⍳II x[a+b+12+c]]
[5]   z←z,⎕AV[4]
    ∇


    ∇ z←II x
[1]   z←⍉⌊(256⊥x)-(256*''ρ(ρx),1)×128≤(1↑ρx)ρx←⍉⎕AV⍳x
    ∇
```

References

[1] Adobe Systems Inc., *Type 1 Font Format*, Addison-Wesley Publishing Co., Inc, 1990.

[2] C Julius Caesar (died 44 B.C.E.), *De Bello Gallico*, I.i.

Finally

Before downloading any fonts to your printer, please make sure that you have the legal right to do so.

N.B. (Production Ed) ... Line-1 may also be entered as:

```
⎕AV[4,115 101 114 118 101 114 100 105 99 116 32 98 101 103 105
110 32 48 32 101 120 105 116 115 101 114 118 101 114,13]
```

Use of Multivariate Techniques for the Grouping Together of Districts. Application in the City of Granada

by Cano Guervos, R.A., Chica Olmo, J. & Hermoso Gutierrez, J.A.
University of Granada (Spain)

Paper given at "APL in Business", Swansea 18-22 July 1994

Abstract[1]

Housing, as an urban property, shares a great number of characteristics of an extremely varied nature. The handling and interpretation of such a quantity of variables makes their synthesized representation necessary. Analysis in Principal Components (APC) is a statistical tool which greatly helps to carry out this task.

In this paper, the main results obtained after analyzing the housing supply in the city of Granada (Spain) are presented, from the point of view of APC. Multivariate statistical contrasts are applied with the aim of confirming the similarity between districts in the city.

1. Aims of the Study

The study has been carried out using a questionnaire from the Department of Real Estate Appraisals of the Andalucia Regional Government. The size of the sample is 299 houses (apartments) up for sale, on which data has been obtained with regard to fourteen variables, see Table 1.

The aims of the study are:

a) To determine which of the fourteen variables best shows up the differences among this group of these houses.

b) To reduce the fourteen variables down to a smaller group, linear combinations of the original ones.

c) To determine which districts present similarities and radical differences with regard to the set of variables under consideration.

Variables

1. Floor.
2. Conservation: high values imply a bad state of repair.
3. Bathrooms
4. Constructed surface area of the apartment (in m^2).
5. Price: that presented on the market by the seller.
6. Quality of the area.
7. Quality of the site.
8. Quality of the building.
9. Quality of the house.

In variables 6, 7, 8 and 9, high values imply high quality.

10. Age.
11. Number of rooms.
12. Outward facing rate of rooms.
13. Price per square metre.
14. Grassed areas: from 1 to 3 points are given if there is a swimming pool, sports areas or garden areas; no points are given if this is not the case.

Table 1

2. Results of the Analysis in Principal Components

2.1. Interpretation of the factors

In Table 2, the factors[2] are presented, classified in decreasing order of variability. In the first part of the study we shall concentrate mainly on the two first factors, since they include simultaneously 51.95% of the total variability of the sample.

The correlations of each of the variables with the first two factors can be seen graphically in Figure 1. By analyzing Figure 1, the significance of the first two factors can be interpreted, keeping in mind the proximity of each of the variables to the factors.

Components or factors	Variance percentage	Accumulated percentage
1	34.409	34.409
2	17.534	51.944
3	10.230	62.174
4	7.477	69.652
5	6.631	76.283
6	6.274	82.558
7	4.211	86.770
8	3.607	90.377
9	2.838	93.215
10	2.283	95.499
11	2.061	97.560
12	1.529	99.089
13	0.746	99.836
14	0.163	100.000

Table 2

First factor (F1):

The characteristics which are most correlated to F1 are: price (-0.87), quality of the building (-0.85), quality of the house (-0.83f), quality of the area (-0.83) and price per square metre (-0.73).

The price variable is the most related to the most discriminating component, which is coherent with the fact that it is a variable which shows a high variation coefficient in the initial data.

If we observe the positioning of the points-variables on the first axis (Figure 1), we can see that the variables with a greater weighting, apart from price, refer to the quality of the construction and the area. The quality of site and the number of bathrooms[3] are found to have a slightly lower influence. Consequently, we can interpret this factor as the "overall quality of the house".

Second factor (F2):

The characteristics which are most correlated to this factor are: number of rooms (0.83), constructed surface area (0.71) and, to a lesser extent, the number of bathrooms. Therefore, the second factor is interpreted as "size of the house".

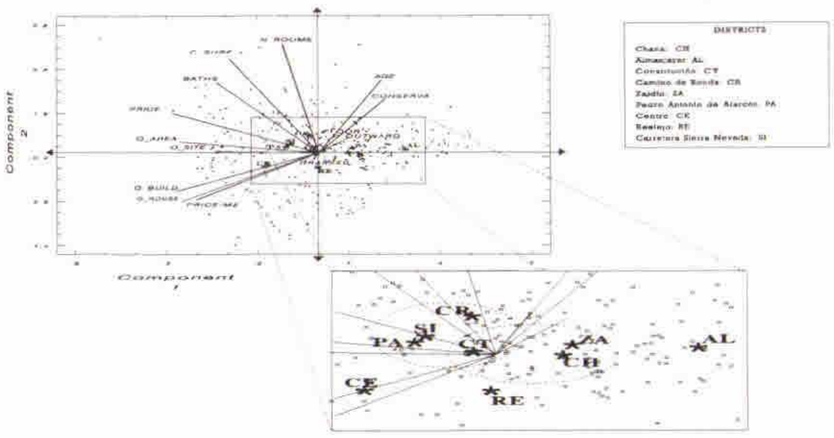


Figure 1.

2.2. Interpretation of the variables correlation plot

The observation of Figure 1 shows the existence of three groups of variables:

A. A first group of characteristics very close to each other and, therefore, highly correlated among themselves: quality of the construction (of the building and of the house) and environmental quality (of the area and of the site). The houses for sale in the sample reflect that construction is carried out with good qualities in urbanistically good areas, and that it is difficult to find high building quality in low environmental quality areas. Furthermore, a significant correlation is noticeable between price per square metre and the previous group of variables, particularly the quality of the area.

B. Another group of characteristics is located in the second quadrant: number of rooms, constructed surface area and bathrooms. The relatively significant correlation between quality of the area and bathrooms should be pointed out here.

C. The third group of variables is found in the first quadrant: age (high) and conservation (bad), which move in the opposite direction to the qualities of construction.

2.3. Interpretation of the individuals scatterplot

In order to complete the APC, we project a set of supplementary individuals: the centres of gravity of each quarter. This centre of gravity represents the average type of house in the district.

A grading of lower to higher quality (right to left) can be seen in Figure 1 for the houses in each quarter: From the Almanjáyar district (AL), which is the most run-down sector of the city, to the Centre area (CE), where the highest environmental and construction qualities are found.

The position of the points with regard to F2 should be studied carefully, since the quality of representation on this axis is only relatively good in the districts of Camino de Ronda (with houses of a greater size) and Realejo (smaller houses).

3. Grouping Together of Districts

We shall now try to point out the possible differences and similarities of houses among the different districts of the city.

In order to confirm or reject the groupings which can be made in sight of Figure 1, various statistic contrasts of a multivariate nature[4] have been achieved.

There are various reasons to justify the use of the principal components instead of the original variables when carrying out the contrasts: one of them lies in the advantage that the contrast can gradually become richer with a greater quantity of information, which is ensured with the gradual introduction of the principal components; on the other hand, making use of some from the original variables would be more arbitrary. Another advantage in using the principal components is that the greater part of the information from the original variables is included in just a few of the principal components.

The T^2-Hotelling test has been used with the aim of comparing pairs of districts with respect to various principal components.

The test used to compare three or more districts with respect to various components is the likelihood ratio test of the multivariate analysis of the variance which, from now on, shall be called ϕ.

3.1. Results of the multivariate contrasts

After applying the T^2-Hotelling test with two components to all the possible district pairs, the isolation of three districts is noticeable: Almanjáyar (AL), Centre (CE) and, to a lesser extent, Realejo (RE). The explanation of the separation of

Almanjáyar (AL) and Centre (CE) from the rest of the districts is clear: the first one is a suburb which, socioeconomically, is considered as run-down; the second has traditionally always been the Centre Business District (CBD).

If the first two components are taken into account (see Table 3), the existence of two large groups of districts is evident: on the one hand, the group formed by Constitución-Toros (CT), Camino de Ronda (CR), Pedro Antonio de Alarcón (PA) and Camino de la Sierra (SI); on the other hand, the group which includes Constitución-Toros (CT), Chana (CH) and Zaidín (ZA) (see Figure 1). The first group of districts can be characterised from a socioeconomic point of view as being middle or upper-middle class, whilst the second group is middle and working class.

If a greater quantity of information is added through the third component, the homogeneity of the first group is lost, whilst the second group remains together (see Table 4). This second group (CT, CH and ZA) is lost when the fourth component is added (see Table 5). After introducing the four components, only the pair formed by Zaidín (ZA) and Chana (CH) is maintained. This grouping shows a strong homogeneity since it is maintained when the first six components are introduced, which include more than 80% of the quantity of information in the sample.

Districts	ϕ exp.	G.L.	ϕ theor.	Group?
CH, CT, ZA	7.836	4	9.488	YES
CT, CR, SI	6.914	4	9.488	YES
CT, PA, SI	3.335	4	9.488	YES
CT, CR, PA, SI	8.452	6	12.602	YES
CH, CT, ZA, RE	24.528	6	12.602	NO
CR, PA, SI	5.930	4	9.488	YES
CT, CR, PA, RE, SI	23.985	8	15.514	NO
CT, CR, PA, SI, RE, CH, ZA	51.581	12	21.029	NO
CT, CR, PA, SI, CH, ZA	32.170	10	18.313	NO

Table 3. Groups of homogeneous districts in relation with two components.

Districts	ϕ exp.	G.L.	ϕ theor.	Group?
CH, CT, ZA	9.978	6	12.602	YES
CT, CR, PA, SI	40.148	9	16.925	NO

Table 4. Groups of homogeneous districts in relation with three components.

Given that the inexistence of significant differences has been accepted with regard to the average behaviour of the Zaidín and Chana districts, the possibility of significant differences between both districts is studied with respect to the dispersal of their data.

The majority of texts about multivariate methods suggest the use of the Bartlett test to compare the variation in two multivariate samples.

Districts	ϕ ó T^2 exp.	G.L.	ϕ ó T^2 theor.	Group?
CH, CT, ZA	15.635	8	15.514	NO
CH, ZA	*1.337*	*4, 36*	*2.633*	*YES*
CT, CR, PA, SI	41.326	12	21.029	NO
CT, PA, SI	26.898	8	15.510	NO
CR, PA, SI	36.044	8	15.510	NO
PA, SI	*6.347*	*4, 77*	*2.490*	*NO*

Table 5. Groups of homogeneous districts in relation with four components.

This test is very sensitive with respect to the hypothesis of multivariate normality (Srivastava and Carter, 1983, p. 333). A more robust alternative process is the Levene test (Levene, 1960), which transforms the original data in to absolute deviations from sample mean or median (using this last one gives even greater robustness). Another possibility is the Van Valen test (Van Valen, 1978), which calculates:

$$d_{ij} = \sqrt{\sum_{k=1}^{p} \left(x_{ijk} - \bar{x}_{jk}\right)^2}$$

where x_{ijk} is the value of the variable X_k for the ith individual in sample j and $\bar{x}_{jk}$ is the mean of the same variable in the sample (the median can also be used, a more robust test being obtained). The test is based on the assumption that in one of the two districts compared there is more variability over all the components.

The Van Valen test, as opposed to the Levene test, has the advantage of being directional, in the sense that it emphasises a greater variation comparing one district with another when this variation points in the same direction for all the variables.

After applying the Levene test to the first six components in Zaidín and Chana, no significant differences are observed with regard to variability (see Table 6).

	Exp. value	**G.L.**	**Theor. value**	**Group?**
Levene 6 components	0.341	6, 34	2.380	YES
Van Valen 6 components	-1.270	39	-1.685	YES

Table 6. Results of the contrasts of Levene and Van Valen

Given that the differences with respect to dispersal in the samples of Zaidín and Chana, although small, point in the same direction, it could be thought that the Van Valen test would help us to detect a greater overall variability in the Zaidín data as opposed to those of Chana. After carrying out the test over the first six components, no significant differences are found either with reference to dispersal that would support said intuition (see Table 6). Therefore, the existence of a great homogeneity is noted among the houses for sale in these two districts, in both terms of average behaviour and variability. The previous analyses confirm the similarity between these two districts which, although geographically distant the one from the other, occupy symmetrical positions with regard to the city map and present common socioeconomic and urbanistic profiles.

4. Bibliography

Cano Guervós, R.A., Chica Olmo, J.M., Hermoso Gutiérrez, J.A. (1993). *Categorización de las Características de la Vivienda en Venta en la Ciudad de Granada.* Estudios de Economía Aplicada, vol. I. VII Reunión de ASEPELT-España. Servicio de Publicaciones de la Universidad de Cádiz.

Levene, H. (1960). *Robust tests for equality of variance.* Contributions to Probability and Statistics. Stanford University Press, California.

Manly, B. (1986). *Multivariate Statistical Methods. A Primer.* Chapman and Hall, New York.

Saporta, G.(1990). *Probabilités Analyse des Données et Statistique.* Editions Technip. París.

Srivastava, M.S., Carter, E.M. (1983). *An Introduction to Applied Multivariate Statistics.* North Holland, New York.

Van Valen, L. (1978). *The Statistics of Variation.* Evolutionary Theory, vol. 4.

Ball, J.M. (1973). "Recent Empirical Work on the Determinants of Relative House Prices". *Urban Studies.*

Dericke, P.H. (1983). *Economía y Planificación Urbana.* IEAL. Madrid.

Lebart, L., Morineau, A., Fenelon, J.P. (1985). *Tratamiento Estadístico de Datos.* Marcombo. Barcelona.

[1] Financed by project PB91-0952 (DGICYT).

[2] Each factor is an artificial and latent variable which is obtained as a linear combination of the fourteen variables under consideration (SAPORTA, 1990).

[3] The number of bathrooms can be generally considered as indicative of the quality of the house.

[4] A brief explanation of these contrasts can be found in Bryan F.J. Manly: *Multivariate Statistical Methods,* Chapman and Hall Ltd., New York, 1986.

Niven Numbers and APL

by Joseph De Kerf

A positive integer or natural number is a *niven number* if it is divisible by its digital sum [1]. For instance, the integer 12 is a niven number as 12 is divisible by 1+2=3, while the integer 11 is not a niven number because 11 is not divisible by 1+1=2. Niven numbers were defined as such by R. Kennedy [1].

Starting with the niven number 1, let *F(N)* be the *N*th niven number. Niven numbers *F(N)* for *N* = 1(1)100 are listed in Table 1.

1	2	3	4	5	6	7	8	9	10
12	18	20	21	24	27	30	36	40	42
45	48	50	54	60	63	70	72	80	81
84	90	100	102	108	110	111	112	114	117
120	126	132	133	135	140	144	150	152	153
156	162	171	180	190	192	195	198	200	201
204	207	209	210	216	220	222	224	225	228
230	234	240	243	247	252	261	264	266	270
280	285	288	300	306	308	312	315	320	322
324	330	333	336	342	351	360	364	370	372

Table 1: ***F(N)*** **for** ***N*** **= 1(1)100**

Niven numbers *F(N)* for higher values of *N* are given below (*cf.* also Figure 1 for *N* = 200(200)1000).

N	*F(N)*	*N*	*F(N)*	*N*	*F(N)*
200	902	2000	12532	20000	166860
400	2000	4000	27168	40000	357210
600	3102	6000	43080	60000	570042
800	4311	8000	60816	80000	804600
1000	5652	10000	80118	100000	1033781

As *F(N)* increases with *N*, the frequency of the niven number decreases when *N* increases, Some aspects of this frequency have been studied by C. Cooper and R. Kennedy [2]. For instance, it was shown that there can exist at most twenty consecutive niven numbers. Further characteristics of such consecutive niven number sequences were studied by B. Wilson [3].

So far, however, no algorithm has been found to calculate *F(N)* directly from *N*. This means that, to calculate *F(N)*, the sequence of positive integers or natural numbers 1, 2, 3,... has to be checked for niven-ness until *N* niven numbers have been found. In other words a loop of *F(N)* cycles has to be executed.

Two APL user-defined functions *NIVEN*1 and *NIVEN*2 for calculating the sequence of niven numbers *F(1), F(2), ..., F(N)* are given below.

```
       ∇NIVEN1[⎕]∇                          ∇NIVEN2[⎕]∇
      ∇Z←NIVEN1 N;I;J                    ∇Z←NIVEN2 N;I;J
[1]    →(N≤0)/0,Z←⍳I←0                   [1]    →(N≤0)/0,Z←⍳I←0
[2]    LAB:J←ρ⍕I←I+1                     [2]    LAB:J←1+⌊10⍟I←I+1
[3]    →(0≠(+/(Jρ10⊤I)|I)/LAB            [3]    →(0≠(+/(Jρ10⊤I)|I)/LAB
[4]    →(N>ρZ←Z,I)/LAB                   [4]    →(N>ρZ←Z,I)/LAB
      ∇                                        ∇

      NIVEN1 20
1 2 3 4 5 6 7 8 9 10 12 18 20 21 24 27 30 36 40 42

      NIVEN2 20
1 2 3 4 5 6 7 8 9 10 12 18 20 21 24 27 30 36 40 42
```

On line [1], it is checked if *N* is negative, a counter *I* is set to 0, and the explicit result *Z* is set to the empty vector `⍳0`. In line `[2]`, the counter *I* is increased by 1 and the number of digits *J* of *I* is evaluated. In line `[3]`, using the function *encode*, the counter *I* is split into its digits and, using the function *residue*, it is checked if *I* is divisible by the sum of those digits, in which case *I* is a niven number. Finally, in line `[4]`, if the counter *I* is a niven number, *Z* is catenated with this counter, and the loop is closed when the shape of the explicit result is equal to *N* (or ⌈*N*). If *N* is negative or zero, the empty vector `⍳0` is returned. If *N* is positive, the vector of niven numbers `F(1), F(2), ..., F(⌈N)` is returned.

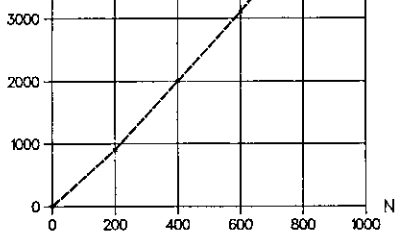


Figure 1: *F(N)* for *N* = 200(200)1000

The difference between *NIVEN*1 and *NIVEN*2 lies in the procedure used to count the digits of *I*. In *NIVEN*1 this is done by evaluating the shape of the format of *I*. In *NIVEN*2 it is done by adding a 1 to the floor of the common logarithm of *I*. To compare the effectiveness of the two procedures, cpu times *T(N)* for both functions have been monitored, and this for *N* = 200(200)1000. Average results in ms for 10 series of 100 executions are given in Table 2 (*cf.* also Figure 2). Differences in performance are negligible.

N	200	400	600	800	1000
*NIVEN*1	266	592	923	1287	1689
*NIVEN*2	268	594	924	1288	1690

Table 2: CPU times *T(N)* in ms for *N* = 200(200)1000

T(N)/N increases very rapidly with *N*, such that for instance *T(N)* becomes about 29.3 *seconds* for *N* = 10000 and about 16.0 *minutes* for *N* = 100000.

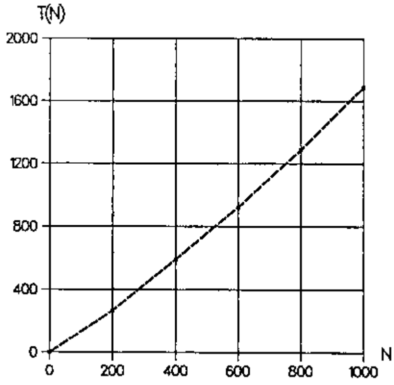


Figure 2: CPU times *T(N)* in ms for *N* = 200(200)1000

In *NIVEN*1 and *NIVEN*2, the count *J* of the digits of *I* and checking if *I* is a niven number is done in two separate lines [2] and [3], this for readability. Performance in cpu times may be improved by substituting lines [2] and [3] by a one-liner. This is done in the user-defined functions *NIVENA* and *NIVENB*

```
         ∇NIVENA[⎕]∇
        ∇Z←NIVEN2 N;I;J
[1]      →(N≤0)/0,Z←⍳I←0
[2]      LAB:→(0≠(+/((⍴⍕I)⍴10)⊤I)|I←I+1)/LAB
[4]      →(N>⍴Z←Z,I)/LAB
        ∇
```

```
       ∇NIVENB[⎕]∇
      ∇Z←NIVEN2 N;I;J
[1]    →(N≤0)/0,Z←⍳I←0
[2]    LAB:→(0≠(+/((1+⌊10⍟I)⍴10)⊤I)|I←I+1))/LAB
[4]    →(N>⍴Z←Z,I)/LAB
      ∇

      NIVENA 20
1 2 3 4 5 6 7 8 9 10 12 18 20 21 24 27 30 36 40 42
      NIVENB 20
1 2 3 4 5 6 7 8 9 10 12 18 20 21 24 27 30 36 40 42
```

Results for the same benchmark as done for *NIVEN*1 and *NIVEN*2 are given in Table 3. Ratios of the results versus those for *NIVEN*1 and *NIVEN*2 are added.

N	200	400	600	800	1000
NIVENA	255	568	886	1236	1624
NIVENB	256	570	888	1237	1624
Ratios	0.957	0.960	0.960	0.960	0.961

Table 3: CPU times *T(N)* in ms for *N* = 200(200)1000

Just as for *NIVEN*1 and *NIVEN*2, differences in performance between *NIVENA* and *NIVENB* are negligible. Improvement of performance of the functions *NIVENA* and *NIVENB* versus the functions *NIVEN*1 and *NIVEN*2 is about 4% for the domain investigated but decreases for higher values of N, such that *T(N)* is still about 28.3 *seconds* for N = 10000 and about 15.6 *minutes* for N = 100000. In summary, performance in cpu times of the user-defined functions given is *very poor* for higher values of N and it would be a challenge to find an algorithm which *drastically* improves this performance.

Programming, calculations and benchmarks have been done on a MicroLine Pentium-S 100, with Dyalog APL/W Version 7.1.2, under Windows 3.11. Benchmarks have been done using the system function ⎕*MONITOR*.

References

[1] R. Kennedy, T. Goodman and C. Best: *Mathematical Discovery & Niven Numbers.* The MATYC Journal, Vol. 14, no. 1, 1980, pp. 21-25

[2] C. Cooper and R. Kennedy: *On Consecutive Niven Numbers.* The Fibonacci Quarterly, Vol. 31 no. 2, May 1993, pp. 146-151

[3] B. Wilson: *Construction of Small Consecutive Niven Numbers.* The Fibonacci Quarterly, Vol. 34 no. 3, June-July 1996, pp. 240-243

[4] *Dyalog APL/W Language Reference – Version 7.* Dyadic Systems Ltd., Basingstoke, 1994

A Note on Cholesky Decomposition

by Tapio Nummi

The aim of this note is to show how to make the Cholesky decomposition effectively by using array operations provided by APL. The Cholesky decomposition for a positive definite matrix A can be written as

$$\mathbf{A} = \mathbf{R}'\mathbf{R}$$

where **R** is an upper triangular matrix. The algorithm is given by

$$r_{1j} = a_{1j} / \sqrt{a_{11}}$$

if $j = 1, 2, \ldots, n$. If $i = 2, \ldots, n$

$$r_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2} \qquad \text{and} \qquad r_{ij} = \left(a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj} \right) / r_{ii}$$

when $j = i + 1, \ldots, n$.

As an example we consider the Cholesky decomposition for a definite 3 × 3 matrix A. This yields

$$\mathbf{R} = \begin{pmatrix} a_{11} / \sqrt{a_{11}} & a_{12} / \sqrt{a_{11}} & a_{13} / \sqrt{a_{11}} \\ 0 & \dfrac{a_{22} - r_{12} r_{12}}{\sqrt{a_{22} - r_{12}^2}} & \dfrac{a_{23} - r_{12} r_{13}}{\sqrt{a_{22} - r_{12}^2}} \\ 0 & 0 & \dfrac{a_{33} - (r_{13}^2 + r_{23}^2)}{\sqrt{a_{33} - (r_{13}^2 + r_{23}^2)}} \end{pmatrix}$$

By using APL the first row of R is

```
R[1;)←A[1;]÷A[1;1]*0.5
```

For the second row we need

```
B←1 1↓A-R[1;]∘.×R[1;]
```

which gives

$$\mathbf{B} = \begin{pmatrix} a_{22} - r_{12}^2 & a_{23} - r_{12}r_{13} \\ a_{32} - r_{13}r_{12} & a_{33} - r_{13}^2 \end{pmatrix}$$

Then the second row of **R** is

```
R[2;]←¯3↑B[1;]÷B[1;1]*0.5
```

For the third row we make new **B**

```
B←1 1↓B-(1↓R[2;])∘.×1↓R[2;]
```

Then by using

$$\mathbf{B} = \left(a_{33} - r_{13}^2 - r_{23}^2\right)$$

we easily compute the third row as

```
R[3;]←¯3↑B[1;]÷B[1;1]*0.5
```

As an APL function

```
      ∇R←CHOLESKY A;B;I;V;N
[1]     R←(ρB←A)ρ~I←1 ⋄ N←1↑ρA
[2]    L:R[I;]←(-N)↑V←B[1;]÷B[1;1]*0.5
[3]     B←1 1↓B-V∘.×V
[4]     →(N≥I←I+1)/L
      ∇
```

The function consists of only one loop and the APL code to iterate is rather minimal. Therefore this function is effective also for large positive definite matrices.

```
      +C←(3,3)ρ1 2 3 2 20 26 3 26 70

1  2  3
2 20 26
3 26 70

      CHOLESKY C

1 2 3
0 4 5
0 0 6
```

Index to Advertisers

All queries regarding advertising in VECTOR should be made to Gill Smith, at 01439-788385, or to apl385@compuserve.com via email.

Submitting Material to Vector

The Vector working group meets towards the end of the month in which Vector appears; we review material for issue n+1 and discuss themes for issues n+2 onwards. Please send the text of submitted articles (hardcopy with diskette as appropriate) to the Vector Working Group via:

Vector Administration, c/o Gill Smith
Brook House
Gilling East
YORK, YO6 4JJ
Tel: +44 (0) 1439-788385
Email: apl385@compuserve.com

Authors wishing to use Word for Windows should contact Vector Production for a copy of the APL2741 TrueType font, and a suitable Winword template. These may also be downloaded from the Vector web site at www.vector.org.uk

Camera-ready artwork (e.g. advertisements) and diskettes of 'standard' material (e.g. sustaining members' news) should be sent to Vector Production, Brook House, Gilling East, YORK YO6 4JJ. Please also copy us with all electronically submitted material so that we have early warning of possible problems.

British APL Association: Membership Form

Membership is open to anyone interested in APL. The membership year normally runs from 1st May to 30th April, but new members may join from 1st August, November or February if preferred. The British APL Association is a special interest group of the British Computer Society, Reg. Charity No. 292,786

Name: ____________________
Address: ____________________
____
Postcode / Country: ____________________
Telephone Number: ____________________
Email Address: ____________________

Category (please tick box) to run from: 1st May ☐ August ☐ Nov ☐ Feb ☐

UK private membership	£12	☐
Overseas private membership	£14	☐
Airmail supplement (not needed for Europe)	£4	☐
UK Corporate membership	£100	☐
Corporate membership overseas	£135	☐
Sustaining membership	£430	☐
Non-voting UK member (student/OAP/unemployed only)	£6	☐

PAYMENT — in Sterling or by Visa/Mastercard/JCB only

Payment should be enclosed with membership applications in the form of a UK Sterling cheque to "The British APL Association", or you may quote your Mastercard, Visa or JCB number.

I authorise you to debit my Visa/Mastercard/JCB account

Number: ____ ____ ____ ____ Expiry date: __ | __

for the membership category indicated above,

☐ annually, at the prevailing rate, until further notice
☐ one year's subscription only

(please tick the required option above)

Data Protection Act:
The information supplied may be stored on computer and processed in accordance with the registration of the British Computer Society.

Signature: ____________________ Send the completed form to:

British APL Association, c/o Rowena Small, 8 Cardigan Road, LONDON E3 5HU, UK

The British APL Association

The British APL Association is a Specialist Group of the British Computer Society. It is administered by a Committee of officers who are elected by a postal ballot of Association members prior to the Annual General Meeting. Working groups are also established in areas such as activity planning and journal production. Offers of assistance and involvement with any Association matters are welcomed and should be addressed in the first instance to the Secretary.

1997/98 Committee

Journal Working Group

Editor:	Vacant Post	see above
Production:	Adrian & Gill Smith	Brook House, Gilling East, YORK (01439-788385)
Advertising:	Gill Smith	Brook House, Gilling East, YORK (01439-788385)

Support Team: Jonathan Barman (01488-648575), Richard and Adam Weber (0121-3546550), Anthony & Sylvia Camacho, Ray Cannon (01252-874697), Marc Griffiths (01653-691745), Bob Hoekstra (01483 771028), Jon Sandles (01904-612882)

Typeset by APL-385 with MS Word 5.0 and GoScript

Printed in England by Short-Run Press Ltd, Exeter

VECTOR

VECTOR is the quarterly Journal of the British APL Association and is distributed to Association members in the UK and overseas. The British APL Association is a Specialist Group of the British Computer Society. APL stands for "A Programming Language" — an interactive computer language noted for its elegance, conciseness and fast development speed. It is supported on most mainframes, workstations and personal computers.

SUSTAINING MEMBERS

The Committee of the British APL Association wish to acknowledge the generous financial support of the following Association Sustaining Members. In many cases these organisations also provide manpower and administrative assistance to the Association at their own cost.

Causeway Graphical Systems Ltd
The Maltings, Castlegate,
MALTON, North Yorks YO17 0DP
Tel: 01653-696760
Fax: 01653-697719
Email: causeway@compuserve.com
Web: www.causeway.co.uk

Compass Ltd
Compass House
60 Priestley Road
GUILDFORD, Surrey GU2 5YU
Tel: 01483-514500

Dyadic Systems Ltd
Riverside View, Basing Road,
Old Basing, BASINGSTOKE,
Hants, RG24 0AL
Tel: 01256-811125
Fax: 01256-811130
Email: sales@dyadic.com
Web: www.dyadic.com

HMW Trading Systems Ltd
Hamilton House,
1 Temple Avenue,
LONDON EC4Y 0HA
Tel: 0171-353 8900
Fax: 0171-353 3325
Email: 100020.2632@compuserve.com

Insight Systems ApS
Nordre Strandvej 119C
DK-3150 Hellebæk
Denmark
Tel: +45 49 76 20 20
Fax: +45 49 76 20 30
Email: info@insight.dk
Web: www.insight.dk

MicroAPL Ltd
South Bank Technopark
90 London Road
LONDON SE1 6LN
Tel: 0171-922 8866
Fax: 0171-928 1006
Email: microapl@microapl.demon.co.uk
Web: www.microapl.co.uk

Soliton Associates Ltd
Groot Blankenberg 53
1082 AC Amsterdam
Netherlands
Tel: +31 20 646 4475
Fax: +31 20 644 1206
Email: sales@soliton.com

Dutch APL Association
Postbus 1341
3430BH Nieuwegein
Netherlands
Tel: +31 347 342 337