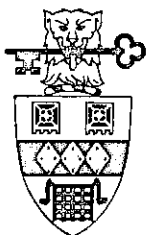


VECTOR

All the *APL* that's fit to Print!

- 16-page Educational Supplement
- APL90: Accepted Papers
- a Tale of Two File Systems
- Greg-y-Nog, and after
- yet more ambivalent letters



*The Journal of the
British APL Association*

A Specialist Group of the British Computer Society

ISSN 0955-1433

Vol.6 No.3 January 1990

Contributions

All contributions to VECTOR may be sent to the 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 (ideally with a photograph of the author and a brief biographical note). Unless otherwise specified, each item will be considered for publication as a personal statement by the author.

Please supply as much material as possible in machine-readable form, ideally on an IBM PC compatible (DSDD) diskette. APL code should be kept separate from the text, and can be accepted as camera-ready copy, or as APL*PLUS/PC workspaces or IBM APL transfer format files.

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 1990

Category	Fee	Vectors	Passes
Non-voting Student	5	1	1
UK Private	10	1	1
Overseas Private	18	1	1
(Supplement for Airmail)	8		
UK Corporate Membership	85	10+	5
Overseas Corporate	140	10+	
Sustaining	360	50+	5

The membership year runs from 1st May to 30th April. Applications for membership should be made to the Treasurer on 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 or Mastercard 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

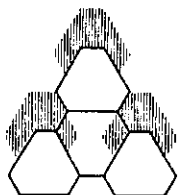
Advertisements in VECTOR should be submitted in typeset camera-ready A5 portrait format with a 20mm blank border. Illustrations should be black-and-white photographs or line drawings. Rates are 250 per full page, 125 for half-page or less (There is a 50 surcharge per advertisement if spot colour is required).

Deadlines for bookings and copy are given under the Quick-reference Diary

Advertisements should be booked with, and sent to, Alison Chatterton, whose address is given with the Index to Advertisers.

CONTENTS

		Page
EDITORIAL:	Adrian Smith	3
APL NEWS		
Quick Reference Diary		5
APL Course Diary		7
General Correspondence		9
British APL Association News	Graham Parkhouse	11
News from Sustaining Members	Alison Chatterton	12
The Education VECTOR	Alan Sykes	17
APL90: Accepted Papers		33
REVIEWS		
APL Product Guide	Alison Chatterton	37
Is APL*PLUS Getting Overweight?	Adrian Smith	49
A Tale of Two File Systems: Myriade for APL*PLUS and AFM for APL2/PC	Adrian Smith	51
RECENT MEETINGS		
Introduction	Adrian Smith	57
The APL Statistics Library		
Conference at Greg-y-nog	Alan Sykes	59
The BAPLA at British Airways	Sylvia Camacho	70
Managing Multicurrency Accounting	Peter Branson	73
APL for Financial Calculations	D. O. Forfar	76
GENERAL ARTICLES		
Symbolic Computation and Recursion	Tony O'Hagan	80
TECHNICAL SECTION		
Hackers Corner: Beware the M61	Adrian Smith	90
Technical Correspondence	Peter Branson et al	91
Error Trapping in IPSA APL	Peter Biddlecombe	102
Problems for APL Buffs	Zdenek Jizba	109
Using the GEM File Selector in APL.68000	John Sullivan	113
A Vector Construction Kit	Adrian Smith	117
A Note on Quad and Quote-Quad	Joseph L.F. de Kerf	122
British APL Association Software Library		
Submission Details / Order Form		124
Index to Advertisers		127



APL PEOPLE

JOBS!

JOBS!

JOBS!

As THE PREMIER APL employment agency, more APL vacancies are filled through us than any other agency.

Established over eight years ago, our understanding of the APL market-place is unrivalled. We always have a good selection of vacancies available throughout the UK.

To find out what we can do for you, contact

Jill Moss on 0225 462602 during office hours
or 0225 333618 eves & w/ends

or write to her at

The Old Malthouse, Clarence St., Bath BA1 5NS

Editorial: Is There Room for APL?

by Adrian Smith

As we move from the Eighties into the Nineties, it is interesting to speculate on the shape of the computer world ten years hence. It is also (just) passing the point where I can claim 10 years' experience of APL, so I suppose I could use the simplest of forecasting algorithms (beloved of the Foreign Exchange community): tomorrow will probably be a bit like yesterday.

In 1979 there were no IBM PC's, few interactive terminals (and most of those were the noisy typewriter sort), almost no colour, practically no graphics. Oh, and there was APL, complete with a shared-variable file system and the ∇ editor. APL did lots of amazing things that nothing else could do (like spreadsheets) and it quickly found a following in the business community. It also spread very slowly into the related worlds of mathematics, Operational Research, and Statistics; as it happens I first came across APL as a tool for building a distinctly complex production planning system: a typical piece of O.R. work.

In 1990, you can tote an IBM PS/2 around on the London Underground; this machine sports an elegant grey satchel, and packs nearly 4 times the punch of the entire CPU complex that Rowntrees owned in 1979. It has (among the spreadsheets and wordy processors and simulation modelling tools) an APL interpreter of significantly greater power than anything available in 1979, and it does rather good colour and graphics. In spite of this huge increase in power and functionality, there remains one rather unpalatable fact: the business community is deserting APL in droves, and at an ever-increasing rate.

So ... is there room in the hard commercial world of the Nineties for APL as a serious professional programming language? If not, where is the language likely to pop up next? Let me offer a few speculations, and maybe (if we keep going that long) the lucky editor of VECTOR 16.3 can have some fun taking them apart:

- the majority of big companies (like Rowntrees and British Airways) will take steps to isolate the use of APL into specialist areas like O.R. They will cease to use it for 'mainstream' systems development, and will progressively eliminate existing APL systems from strategic areas of the company. This process is likely to take longer than 10 years, but by 1999 I would expect to see almost no new APL systems in these areas, and many current flagships will have been decommissioned.

- specialist areas who are big enough to resist the drift (I reckon you need a department of at least 6 to support continued APL expertise) will continue to get the support of top management, and will continue to irritate I.T. professionals well into the next millenium. They will get off the mainframe or will be (gently but firmly) kicked off. PCs are already an attractive alternative for APL; they become that much more so when you need to work under cover!
- the slow spread of APL awareness in the mathematics community will continue, and will pick up significantly as the next generation of computers is made available to our schools in the mid-nineties. I suspect that when we look back on the development of APL from a long way ahead (say 2020) we may see a steady growth in the use of APL by the mathematicians, masked by the huge blip of its temporary foray into the world of commerce.
- APL will begin to be appreciated for its internationalism. A very large percentage of the world's mathematics takes place to the East of what we used to call the Iron Curtain. In the Soviet Union they at least use a familiar system of writing, although some of the symbols differ; in Japan even the system is different. I know that I can read Japanese APL as easily as I can read American APL or German APL; isn't it time someone told the Soviets about it?

I suppose it would be equally valid to ask 'is there room for Fortran?' or 'is there room for Cobol?'. If you look at commercial programming, you will again see a rapid decline in conventional languages, as the power and flexibility of packages converges on the requirements of the 'average' company. However I cannot see the world's physicists giving up Fortran, and there will always be the odd hole for Cobol code to nest in.

I hope and believe that APL is quite different; that it is in some peculiar way *orthogonal* to the Fortrans and Cobols and Cs of this world. However hard the I.T. professionals try to stamp us out, we APLers are always likely to pop out again somewhere quite unexpected; I am reminded of the biology of the Common Horsetail, which (according to Stephen Young in a recent New Scientist):

"Aeons of evolution have have left with an unparalleled degree of resistance. Throughout last summer our tarmac contractor returned to our drive - chlorate or Paraquat in one hand, sledgehammer in the other - at least half a dozen times. On each occasion he was routed by this most virulent of weeds.

Nothing short of a chemical holocaust will kill it. If you drench the visible stems with a systemic herbicide - one that spreads through any normal plant - the horsetail shrugs it off, refusing to fall for so simple a ploy. While the docks and dandelions die back, the horsetail comes back for more. In any case, its tissues are resistant to many commonly used herbicides, presumably because its biochemistry departs from that of the common herd."

A Happy New Decade to all you *Equisetum arvense* out there.

Quick Reference Diary

Date	Venue	Event
12 January 1990	London	BAA Meeting
16 February	London	BAA Meeting
23 March	London	BAA Meeting
20 April	London	BAA Meeting
8th June	London	BAA Annual General Meeting
13th - 17th August 1990	Denmark	APL90: APL for the Future

British APL Association meetings are to be held at the Royal Over-Seas League, Park Place, near Green Park Tube station. Meetings begin at 2.00pm and close at 5.30pm.

APL PROGRAMMER/ANALYST

- Around 3 years APL ● Maths/Stats/OR Background ●
- Development/Support on large PC based decision support systems ●
- Worldwide 'Blue-Chip' Users ● APL*PLUS/PC and APL*PLUS/II ●

This is a rare opportunity to join our small, enthusiastic software team. You will often be working on your own initiative and will quickly see the results of your efforts, which will involve extensive contact with users. Its hard work, but a great job with rewards to match.

Send your C.V. to Gareth Brentnall at Mercia Software Limited.

MERCIA
Software Limited

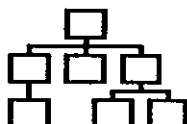
Aston Science Park, Love Lane, Birmingham B7 4BJ
021-359 6086

A single source for APL



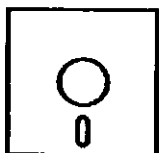
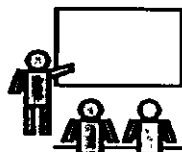
Advice

Design



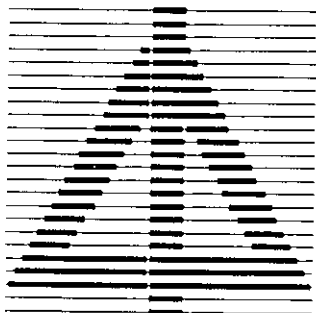
**Application
Development**

Education



APL software

Support



COCKING & DRURY LTD.

THE APL PROFESSIONALS

180 Tottenham Court Road

London W1P 9LE

Tel: 01 - 436 9481

(From May 1990, our STD code will be 071)

APL Training Courses for 1990

compiled by Adrian Smith

Dates shown cover the period from January to November 1990. For confirmation of dates and/or further details please contact the vendor directly.

Level	Company	Days	Dates
Beginners	Cocking & Drury	3	16 Jan, 20 Feb, 24 April
	MicroAPL	1	11 Jan, 8 Mar, 10 May, 14 June, 2 Aug, 4 Oct
APL*PLUS/PC	MicroAPL	1	8 Feb, 29 Mar, 31 May, 19 July, 13 Sept, 1 Nov
Intermediate	MicroAPL	1	15 Mar, 17 May, 5 July, 9 Aug, 11 Oct
APL*PLUS II conversion	MicroAPL	1	15 Feb, 5 Apr, 7 Jun 23 Aug, 8 Nov
Advanced	MicroAPL	1	22 Mar, 24 May, 12 July, 6 Sept, 18 Oct
(Nested Arrays)	Cocking & Drury	2	13 March, 15 May
System Design	Cocking & Drury	4	20 March
Statgraphics	Cocking & Drury	2	13 Feb

Mercia Software Ltd have asked us to note that they no longer offer a regular program of public courses in APL and related subjects.

Dates for Future Issues of VECTOR

	Vol.6 No.4	Vol.7 No.1	Vol.7 No.2
Copy date	1st March 90	1st June 90	7th Sept 90
Ad booking	15th March 90	16th June 90	21st Sept 90
Ad Copy	25th March 90	30th June 90	30th Sept 90
Distribution	April 90	July 90	October 90

dyalog APL

for 386 PCs under MS-DOS

Dyalog APL for 386 PCs under MS-DOS is a complete implementation of Dyadic's popular APL for Unix systems, AND fully exploits the power of a DOS 386-PC.

JUST LOOK AT THE FEATURES . . .

- *An exciting new Screen Manager in which the contents of the screen are 'shared' with a nested system variable □SM (Screen Map).
Facilities include:*
 - Pop-up windows by localisation of □SM
 - Scrolling (vertical, horizontal, linked)
 - GENUINE numeric fields
 - Date fields
 - NESTED forms
- Fully integrated window-based Session Manager, Editor and Debugging Environment.
- Compatible with the Draft ISO standard for APL
- Huge Workspaces (16Mb and MORE!)
- Virtual Memory Support (using on-chip demand paging)
- Full 32-bit processing speed
- *APL2 features including:*
 - Nested Arrays
 - Defined Operators
 - Selective Specification
- *Commercial features including:*
 - FMT
 - Component Files
 - Error Handling
 - "AP124-like" Screen Manager
- External Variables
- Interface to GSS*CGI for fast device-independent graphics
- C Interface

. . . THE BENEFITS SPEAK FOR THEMSELVES

Requirements: 386/486-based PC or PS/2, 640K DOS Memory plus 2Mb RAM, EGA or VGA (colour STRONGLY recommended), 80387 coprocessor STRONGLY recommended, DOS 3.3 or later.

*For further information contact:
Dyadic Systems Limited*

Park House, The High Street, Alton, Hampshire, GU34 1EN
Telephone: (0420) 87024 Fax: (0420) 89886 Telex: 85881

GENERAL CORRESPONDENCE

The VECTOR working group welcomes correspondence on any topic affecting the APL community. All letters should be addressed to the Editor, and should indicate whether they are for the general or technical section. (Letters containing APL code will normally be classed as 'technical'.) The editor reserves the right to edit any letter unless the writer states that it is to be published in full or not at all.

From: Jill Moss

15th December 1989

I would like to complain about an advertisement which appeared in Vector Vol.6 No.2, in which a certain other agency claimed to be the market leader in the APL arena. This is most obviously untrue and I believe contravenes the A.S.A.

As most people will know, APL People is the real market leader in placing people with APL skills. We have been placing people in APL jobs for over eight years now, and probably nine out of every ten vacancies are filled through us. We have placed people in jobs not only throughout the UK, but also abroad, including in Canada, Belgium, France, Luxembourg, Sweden and Spain. We have contacts with companies using APL in many other countries as well.

For VECTOR to print such a misleading advertisement is not only unfair on us, but also unfair on people looking for jobs and staff. I really feel that an official apology is due here, together with an assurance that, in future, the Editor will check advertisements to be sure that the advertisers are not claiming to be something which they most clearly are not!

Jill M Moss
Managing Director
APL People Ltd.

Future visitors to Brook House may like to note the scorched patch on my doormat! I am sorry jill got so upset about this, but I am really in no position to verify claims made in either advertisements, sustaining members news, or correspondence such as the above.
Ed.

From: Andrew Wiggins

7th December 1989

The report of the I-APL board (Vector - October 1989) was very interesting. What I did find surprising was the statement, by Ed Cherlin, that there was a need to generate cash flow. I also found it surprising that the board thought "... that I-APL should move towards being a more commercial enterprise ...".

In my experience, cash flow and commercial enterprises go hand in hand with selling, and that selling usually results in sales. Any commercial enterprise that doesn't sell doesn't generate cash, and eventually disappears without trace.

The point of my letter is this: earlier I ordered an upgrade of I-APL, together with some other library diskettes. To date it still hasn't arrived. All that I have received is a note that the library has not operated this year.

Wither APL nem con

Andrew Wiggins
Manager - Computer Systems Development
Lombard House
3 Princess Way
Redhill
Surrey RH1 1NP

British APL Association News

from Graham Parkhouse

At the November committee meeting we heard that Jonathan Barman is willing to take over from Adrian Smith as editor of Vector. He will need no introduction to many of you as he has been prominent in the APL community for a long time, for many years with Cocking & Drury and now with his own consultancy.

He has been a member of the Vector working group from the beginning and used to edit the technical section with David Ziemann until Vol. 4. We are still looking for a new honorary secretary for the association, and I would welcome enquiries from anyone who would like to know what they would be letting themselves in for before deciding whether they would be willing to go for it.

Outstanding Achievement Award

Please send your nominations for this award to me, Graham Parkhouse, by 30th April 1990. The first mention of this award is in Vector Vol. 3 No. 4 where it states "any person or group that has achieved some outstanding task with or for APL or performed some outstanding service to the APL community or in the promotion of APL to the non-APL world may be a suitable candidate". The prize of a gilded glass apple, held for the year, and cheques for £500 have been won by John Scholes in 1988 for the development of Dyalog APL and jointly by Delia Hickson and Graeme Davison in 1989 for the development of AIMS, British Airways' suite of management information software.

Who will it be in 1990? Each nomination should include a citation and reasons why the achievement is outstanding in no more than one A4 page of ordinary type, signed by two members of the British APL Association. Nominations will be judged by a sub-committee and the result announced at the AGM on Friday 8th June 1990.

The sub-committee reserves the right not to make an award should the nominations not be considered to be sufficiently outstanding.

News from Sustaining Members

Compiled by Alison Chatterton

Intelligent Programs Limited

The newest sustaining member of the British APL Association has been operating for over 10 years in continental Europe and for 6 years in the UK. Our first news as a sustaining member is a brief summary of our company history.

IPL was set up in its current structure in 1989, following the reorganisation of a group of associated companies in several European countries. The company was founded in Sweden in 1979. Since then, we have undertaken projects and installed systems throughout the world, in 9 countries and 4 continents.

We have always specialised in developing systems for the insurance and financial markets, and the majority of our business is in this field. Our systems are hybrids written in APL, Assembler and C. Use of these languages enables us to develop applications with the flexibility of APL and the efficiency of more conventional languages. The systems are designed to offer end-users a modern, user-friendly interface. IPL's other activities include consultancy work.

The company is rapidly expanding. Several new staff have joined in the past 6 months, and we expect to be announcing shortly a major new systems contract. We are pleased to support the British APL Association as a sustaining member, and we believe our systems demonstrate to wider audiences the power and flexibility of APL.

For further information about IPL, please contact Alastair Charatan, Marketing Director on 01-481-4813.

HMW Computing Limited

HMW Computing Limited continue to develop and market "4XTRA"; our front-end Foreign Exchange Dealing and Position Keeping system. The "Computers in the City" exhibition at the London Barbican Centre generated increased interest in the product. We have a number of exhibitions early next year, and welcome Vector readers. Please call us for an invitation. On the hard work and coding front, 4XTRA was installed in a major client's site on the 11th of December: well done the HMW development team!

Our employment agency for the APL community continues to attract permanent positions with a variety of clients throughout the UK. Currently we have many vacancies for APL technicians from Juniors through to Consultants, with attractive salaries and interesting work content.

To find out what we can offer, please call Val Lusmore on 01-353 4212. If you would prefer to phone out of office hours, Val's home number is 07618-727

APL People Limited

As THE PREMIER APL EMPLOYMENT AGENCY in the UK (if not the world!), APL People placed a record number of people in jobs in 1989. We plan to exceed that in 1990!

We invariably get to know about EVERY APL vacancy in the UK, so we always have a good selection of interesting positions available for people with APL skills. These include vacancies for junior programmers, technical experts, consultants, team leaders and even project managers. Our reputation for getting results is fast spreading abroad as well as here.

We also intend to expand our tailor-made training service for companies, offering courses conducted on the company's site and structured to suit the needs and abilities of the trainees.

Delia Marlow has joined Jill Moss in the Bath office, where she will be helping Jill with the ever expanding agency business.

So, whether you're looking for staff, or looking for a new job, call Jill or Delia on 0225 462602 (0225 333618 eves and weekends)!

Dyadic Systems Ltd

Dyadic is now shipping Release 2 of the DOS/386 implementation of Dyalog APL. Release 2 contains a major enhancement to the Tracer, and some significant performance improvements.

In Release 1 the Tracer allowed you to single step through the function or defined operator, with the State Indicator represented by a stack of overlapping windows displaying your APL code. In Release 2 there is an additional 'naked trace' facility.

Now, if an error occurs while you aren't tracing, you can just press <TRACE> and the system will recreate the trace stack. You can then use all of the facilities

of the Tracer to debug your application around the point where it has failed. Typing `→□LC` resumes execution and stops the Tracer, so you can continue once you have fixed the code. Dyadic is also developing facilities to display, trace and edit variables in pop-up windows on the screen. Watch this space for further news!

Performance improvements have been made in dyadic iota and grade, and in the arithmetic primitives `+`, `-`, `×`, `⌊`, and `⌈`. All applications will benefit, but those that perform a lot of arithmetic on large arrays will gain most. Some fairly typical actuarial applications run about 30% faster under Release 2 than under Release 1.

Dyadic is providing the upgrade to Release 2 free of charge to all registered users.

Dyalog APL Release 5.2 has been ported to the Sequent, a large multi-processor system that is also marketed by Prime and Unisys.

Dyadic has started work on an implementation of Dyalog APL for IBM's AIX PS/2. This is a powerful but reasonably priced multi-user Operating System which runs on any 386-based PS/2 such as a model 70 or 80. Dyadic considers this to be an ideal platform for a small multi-user APL application.

Cocking and Drury Ltd

Last December saw the first run of our "APL in the 90s" seminar, a detailed look at the future of APL. The half-day seminar was directed at selected clients, but is now available in-house. Topics covered include the use of APL in the UK, how strategic is APL, vendor attitudes to APL, APL development strategy, APL and SAA, APL and OS/2, and CASE tools and 4GLs.

The seminar expresses our confidence in APL and demonstrates how the language is being used more than ever, particularly by organisations which require solutions to complex business problems. The principal speaker is Romilly Cocking. Please contact Beverley Satterley for further details.

We are planning a new workshop entitled "Choosing the right tools for implementing Decision Support Systems". This two-day practical event is aimed at middle managers, with a slant on using existing tools to implement applications. Again, for further details please contact Beverley Satterley.

A number of our clients have expressed interest in Cocking & Drury offering a comprehensive "APL Facilities Management" service. This service would

encompass the support, maintenance and enhancement of all the customer's APL applications, and would also include the migration of such systems to other environments, if necessary. Contact Romilly Cocking for more information.

On a more general note, the company has recently been looking at Object Oriented Programming languages, and Smalltalk in particular. We would be interested in exchanging information with anyone who has gained application experience using this technology.

Release 9 of the famous APL*PLUS PC System contains a number of facilities which may be new to many users. One such tool is the User Command Processor, which effectively allows you to write your own APL system commands. These commands start with a right-bracket -] rather than a right-parenthesis -). The commands are written in APL, and can be invoked from any APL workspace. No workspace resident code is required in order to use the command. This means that the code of commonly used tools and operations can be centralised, and yet is available to all workspaces and users.

Release 9 comes with a large number of pre-written User Commands, including tools to help you list function globals, locals, unreferenced locals and labels, produce workspace listings, search the workspace for specified strings, package a function and all its sub-functions, and much more. This high-productivity tool has been available on APL*PLUS Mainframe for some time, and we are pleased to see it in the PC environment.

Other Release 9 features include a fast full-screen numeric editor, an automatic symbol table clean-up option, and a new print spooling option which lets you continue work while `□ARBIN` is sending a report to the printer.

Release 9 also comes with a complete replacement set of documentation, with a much improved easier to read tyeface.

MicroAPL Ltd

MicroAPL Ltd. started trading in January 1980 and to celebrate our 10th birthday we are planning a series of events for 1990. We hope to start off with a bumper edition of MicroAPL News, although avid readers of Vector will know how difficult it is to produce these things quickly! Without pre-empting our own newsletter, I think it is fair to say that in our first 10 years we have done more than most to spread the knowledge of APL to a wider audience, and also to reduce the costs of using APL. A large number of our many thousands of users of APL.68000 for machines such as the Sinclair QL, Apple Macintosh, Commodore Amiga and Atari ST are first time APL users. Our policy of issuing

free runtime licences for APL.68000 has also made it feasible to distribute APL-based packages at a reasonable cost.

Version 7.30 of APL.68000 for the Commodore Amiga is now on general release. Clients should have received details already, but if not please contact MicroAPL for upgrade order forms. The upgrade pack includes reprints of both the Amiga-specific booklet and the full APL.68000 manual.

In the Unix environment, version 7.30 of APL.68000 is now available on 68000-based Sun computers, the HP9000 series and 68000-based Bull computers.

We are currently starting work on the upgrade to the Atari ST version of APL.68000 and encouraging users to write in with upgrade requests. We hope to announce a release date for the upgrade in the near future.

MicroAPL's consultancy section finished 1989 with record levels of business and we intend to do better in 1990. Our training activities now include courses in APL*PLUS II and we will be shortly introducing APL2/PC training courses. If you would like an up to date training timetable, please contact us.

THE EDUCATION VECTOR

January 1990

Editor Alan Sykes

This Education Vector has been distributed free to everyone on the I-APL mailing list. It is reprinted from VECTOR Vol.6 No.3. VECTOR is the Quarterly Journal of the British APL Association; for more information about the British APL Association, contact the Secretary: Graham Parkhouse, Dept. of Mech. Eng., University of Surrey, GUILDFORD.

Contents

Editorial	Alan Sykes	18
Puzzle and Problem Corner		20
From Coin Tossing to the Weather	Alan Sykes	24
I-APL News	Anthony Camacho	31
I-APL Order Form	I-APL Ltd.	32

Editor:
Dr Alan Sykes,
Dept of Management Science and Statistics,
University College of Swansea,
Singleton Park, SWANSEA SA2 8PP

Editorial

by Alan Sykes

Welcome once more to Education Vector. After a brief respite, I am back at the keyboard again. (In fact, as an applied statistician, I seem to spend most of my time at what I consider to be the least tuneful of all the keyboards I use!)

Thanks are due to Anthony Camacho, who stepped smartly into my shoes for the last edition, as my spare time had been taken up with organising a conference of statisticians and APL professionals at the University of Wales Conference Centre at Gregynog.

This edition is very much in the mould of its predecessors. I must confess that I enjoy the opportunity of trying fresh approaches to the problems of understanding randomness. (Don't tell me, you'd guessed that already!) This time I've tried to illustrate one of the most important aspects of a statistician's job - namely to look for evidence that might invalidate the current model being used, and if such evidence is found, to construct a new model that overcomes the objections to the original one. This principle, of course, is very much the spirit in which all scientific progress is made.

In addition, I have included a section on some of the puzzles which have appeared in past Education Vectors and the responses that I have had to them. It is very flattering, as well as heartening, that I receive letters from many different people, though I must confess I would be even more pleased to have a greater number of letters from teachers and users of APL in education. One such letter comes from Walter Spunde in Australia who has written at length on how he has used APL to teach linear algebra. I hope to include this contribution in the next edition.

At the international APL conference in New York that I attended in August, it was interesting to see that vendors of APL interpreters are increasingly making their product available to the general public. Of course you don't get everything for nothing, but enough to be very useful. I refer in particular to two pieces of software that are now available.

First is Tangible Math that is produced by I.P. Sharp, and comes with a small explanatory booklet plus disk for IBM PCs. Secondly the recently launched PC APL2 interpreter has been made available by IBM in a form called TryAPL2. This

interpreter is what is called a second generation APL interpreter - hence the '2' in its name. This means that there are many more interesting features available which are not possible with I-APL or any other first-generation APL interpreter. Consequently, the time is ripe for some explanation as to just what this all means for those who are relative newcomers to APL.

In a future edition, I hope to include more details of these products and how to get them. In the meantime, I trust that you will find something of interest in the subsequent pages.

Letter to the Editor

From: R Weber

13th December 1989

Reading the two recent Vectors (July and October), I noted a constant cry for introducing APL in schools. May I make a few points:

1. The problem with schools is that teachers, even if they show interest in APL, cannot easily get the support from Heads because they cannot convince them of the use of APL as a teaching tool, or of the importance of the language.
2. It is still difficult in education to get away from using the BASIC language, mainly due to teachers' convenience. I teach both Basic and APL in Further and Higher Education, and find APL far easier to put over in the short time available (often 12 hours of programming only) with students being able to advance much further than in Basic.

The ability to combine APL with word-processing and spreadsheets is a significant advantage, and should be presented in any educational advertising of APL.

3. I find that the current trend in education is to limit programming, and expand application usage. This is resulting in a stagnation for teachers and lecturers, like myself, with the result that we are being left behind in APL with almost no change of catching up.

To overcome these problems, I would like to suggest the following:

That Local Educational advisory groups be set up. (By local I mean in each town where this is possible.)

That these groups have two objectives: firstly, to go to local schools with a presentation package that would show the full scope of APL (or I-APL) as a teaching tool. It may also be beneficial to extend this to FE colleges. Secondly, to act as a local help group to APL users.

These groups could evolve under the guidance of the Education section of the Association. This can only succeed with the full backing of the Association: the Association, if it is to encompass the education sector, must have a more local approach. We are not able to take part in London meetings due to both financial and time restrictions.

For my part I am very willing to help any local teachers wishing to start in APL, and would appreciate contact with other local APLers to form such a group. I have tried to get teachers interested in APL, but without a more organised approach, it is a losing battle.

I am sure the Association has the material already; it just needs to be put into a 'sellable' package. Using the adage "if the mountain won't come to you - you must go to the mountain", I hope the Association will support and act on this.

Richard Weber

Doncaster Metropolitan Institute of Higher Education
Waterdale
Doncaster DN1 3EX

Puzzle/Problem Corner

Since the last time I put pen to Education Vector paper, I have had a number of letters which gratifyingly inform me that Education Vector is read, by a wide variety of people, spread through many countries. This issue's puzzle/problem corner therefore concentrates on reporting back to you on the many contributions that I have received. Thank you to all those who took the trouble to write to me.

Marks to Grades (Education Vector January 1989)

My first puzzle corner referred to the problem of converting marks into grades and I have already referred to some of the contributions I received on that problem. However, I was delighted to receive one letter that came just too late for me to include then: it comes from Mr McLean in Dunoon, Scotland. Mr McLean, at the time of writing, had had a copy of I-APL for only a few weeks and although his answer to the problem is not as slick as those sent to me by the professionals, it was just as effective and shows what can be done with APL in a relatively short period of time. The solution presented went like this.

```

MARKS←?50 8ρ100
A←MARKS≥70
B←(MARKS<70)∧MARKS≥60
C←(MARKS<60)∧MARKS≥50
D←(MARKS<50)∧MARKS≥40
E←(MARKS<40)∧MARKS≥35
F←MARKS<35
SUM←A+(2×B)+(3×C)+(4×D)+(5×E)+(6×F)
GRADES←'ABCDEF'[SUM]

```

I do hope that readers such as Mr McLean will continue to send in their ideas, reports on their experiences with APL, and I apologise that I am often not too good at replying quickly!

The Card Collecting Problem (Education Vector April 1989)

This was a problem on simulation, to determine the average number of cards that need to be collected in order to get one of each of the set, if the cards received are randomly chosen from the set (I took six in the set). Claude Henrion from Monteiller, Saviese, supplied a solution. The problem with such a simulation is that APL is best at dealing with fixed size arrays, rather than using algorithms of the form 'do something until some criterion is satisfied'. With the card collecting problem, we don't know in advance how many cards are going to be needed, and any algorithm which generates a card at a time, checks to see whether the full set has been achieved, and carries on if not, will be very slow when written in APL.

Claude's solution is to be almost certainly sure of having sufficient cards by taking 66 (11 times the number of cards) at a go. The solution is quite heavy going for newcomers to APL, so I present his one-liner and then show how it works.

```

[1] ∇ R←N CARDS C
    R←+/∨\ϕ∨/∨<\(1C)∘.=?(N,11×C)ρC
∇

```

The expression `5 CARDS 6` then yields five simulations of collecting all six cards. How does this work?

Suppose for example we had only three cards to collect and that one simulation of 6 cards gave the numbers X equal to 1 2 1 2 3 1 - now let's pick up the function to the left of the `?` using X as a substitute for `?(N, 11×C)ρC`

```

      (13)°. = X+1 2 1 2 3 1
1 0 1 0 0 1
0 1 0 1 0 1
0 0 0 0 1 0

```

```

      <\(13)°. = X      (performs the operation 'less than'
1 0 0 0 0 0          accumulatively replacing all
0 1 0 0 0 0          after the first 1 by 0)
0 0 0 0 1 0

```

```

      ∨∧<\(13)°. = X   (performs the logical 'or' down the cols)
1 1 0 0 1 0

```

At this stage, what we need to do is to find out the index position of last 1 to occur, assuming that this last 1 represents the last card required to complete the set.

```

      ϕ∨∧<\(13)°. = X
0 1 0 0 1 1          (reverses the row)

      ∨\ϕ∨∧<\(13)°. = X
0 1 1 1 1 1          (sets all numbers after the first 1 to 1)

```

Hence finally summing this gives the value 5 which is where the last card to complete the set of 3 in 1 2 1 2 3 1 occurred.

Thank you, Claude, for your solution. It shows just what can be done with the useful array manipulative tools that APL provides. However, from a pure computing point of view, it has to be said that ordinary languages which are more geared to recursive steps are more suited to such calculations! Unless of course you, the reader, can do better!

Numbers and Bases (Education Vector July 1989)

In my article on numbers and bases, I looked at whether the digits of a number could be reversed when that same number was represented using a different base. John Sullivan, who is Treasurer of the British APL Association, supplied some nice examples of his exploration of the same field which he had done in response to an article in Computer Weekly. The answer to the question - what bases from 10 to 100 give a reversal for 2 digit numbers using digits 1 to 9 only - is the bases 10 13 16 19 22 25 28, and wait for it, 37 46 55 64 73 82.

John points out in his letter that an upper bound for the bases that need to be tested when exploring a possible reversal of N digits is given by

$$\lfloor (10 \pm 10^{-1} \pm N) \div N - 1 \rfloor$$

which for N equal to 5 gives the answer 17. John also gives a neat function *BASE* to calculate the coefficients of the linear equation used:

$$\begin{array}{l} \nabla R \leftarrow N \text{ BASE } B; \square IO \\ [1] \quad \square IO + 0 \\ [2] \quad R + (B * \phi \uparrow N) - 10 * \uparrow N \\ \nabla \end{array}$$

... so that, for example, 2 *BASE* 16 gives 255 6⁻99. Using *BASE*, John investigates three digit numbers by using:

$$Y \leftarrow CP \ 9 \ 9 \ 9 \\ 10 \uparrow Y [; (0 = (3 \text{ BASE } B) + . * Y) / \uparrow 9 * 3]$$

...with the following results:

base reversible numbers:

9	544	
13	477	
14	438	
16	173	
19	144	288
21	155	219
28	169	

John's energy for APL being inexhaustible, he goes on to look at the 4 digit case which reveals the bases 7 (6112) 16 (1415) and 19 (1277 and 1749). Thank you John!

Since this puzzle section is somewhat longer than usual - my puzzle for this edition is simple - please submit your puzzles for inclusion in the next Education Vector!

From Coin Tossing to the Weather

by Alan Sykes

In this article, we take another random walk that starts from simple counting and coin-tossing and ends with a model for rainfall! Consider the following set of data:

STATES

```

2 2 2 2 1 1 1 1 1 1 2 2 1 2 2 1 1 1 1 1 2 2 2 2 2 2 2 1 2 1 1
  1 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2
  1 2 2 2 1 2 2 2 1 1 2 2 2 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1
  2 2 2 2 2 2 2 2 2 2

```

If we interpret 1 as 'heads' and 2 as 'tails' we could have a sequence of heads and tails. As such, the first thing we might do is to count how many heads and tails there are:

```

  1 2 COUNT STATES
36 64

```

So the first observation is that if the data did come from tossing a coin then it would appear to be a biased coin with something like a chance of 1 in 3 of giving a heads. If you look closely at the data however, you may notice that there appears to be tendency for a head to follow a head and a tail to follow a tail - rather more so than one might expect if the data had come from tossing a coin. How might we investigate whether this is indeed the case?

One way is to count in more detail - instead of counting just the number of heads and tails, we count the number of times a heads is followed by a heads, and similarly for the other three possibilities. We call this counting the transitions from states heads/tails to states heads/tails. How do we do this in APL?

First we construct a 2 by 99 matrix of all the transition pairs - each column of the matrix denoting a transition. This is done by rotating *STATES* one place to the left, using $1\phi\text{STATES}$, forming the matrix, and dropping off the last column. Here are the steps illustrated with $S\leftarrow 1\ 2\ 2\ 1\ 2$

$$1\phi S$$

$$2 \ 2 \ 1 \ 2 \ 1$$

$$S, [0.5]1\phi S$$

$$1 \ 2 \ 2 \ 1 \ 2$$

$$2 \ 2 \ 1 \ 2 \ 1$$

$$0^{-1} + S, [0.5]1\phi S$$

$$1 \ 2 \ 2 \ 1$$

$$2 \ 2 \ 1 \ 2$$

Now all we need to do is to compare each column with the possible transitions 1 1, 1 2, 2 1, 2 2. To do this we form a 4 by 2 matrix:

$$M \leftarrow 2 \rho \begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 1 & 2 & 2 \end{pmatrix}$$

and use the inner product (think of matrix multiplication) constructed by comparing a row and column with '=' and then using '^=' or 'and':

$$M \wedge . = 0^{-1} + S, [0.5]1\phi S$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

The resulting matrix is then easy to interpret. The second row, for example, is 1 0 0 1 because the first and last transitions are from state 1 to 2. Hence the appropriate transition count function can be derived by summing the rows of the boolean matrix, and re-shaping as a 2 by 2 matrix so that the (i,j)th element represents the number of transitions from state i to state j.

Hence we have derived the function *TRANCOUNT*.

$$\nabla R \leftarrow \text{TRANCOUNT } S$$

$$[1] \quad S \leftarrow 0^{-1} + S, [0.5]1\phi S$$

$$[2] \quad R \leftarrow 2 \ 2 \ \rho + / (4 \ 2 \ \rho \ 1 \ 1 \ 1 \ 2 \ 2 \ 1 \ 2 \ 2) \wedge . = S$$

$$\nabla$$

Applying *TRANCOUNT* to the original data set:

$$\text{TRANCOUNT STATES}$$

$$23 \ 13$$

$$13 \ 50$$

And as we expected, from a mere glance at the data, the diagonal frequencies predominate, corresponding to transitions from state to the same state. Of course, we do not have any firm foundation as yet as to whether such a frequency transition matrix is so different from the sort of matrix we could have achieved using (biased) coin tossing that we really are forced to reject our coin tossing model for something more general.

So the next stage is to define some procedure that measures deviations of observed counts/frequencies from expected frequencies. This is the function of the famous (infamous?) chi-square statistic, that takes observed frequencies O , and expected frequencies E , calculates $(O-E) \times (O-E) \div E$, and sums the result.

What do we take as our expected frequencies for the 2 by 2 matrix of transition counts, assuming that the coin-tossing model is appropriate? Well, 36% of the states were heads and so we would estimate the probability of heads as .36 - call this p . If the coin-tossing model is appropriate, then out of 99 transitions, we would expect $99 \times p \times p$ to be heads followed by heads, because the probability of heads followed by heads is just $p \times p$. (The result of one toss is independent of the result of another toss and so the probability of the joint result is the product of the probabilities.) Similarly for the other possibilities, the expected frequencies are $99 \times p \times (1-p)$, $99 \times (1-p) \times p$, and $99 \times (1-p) \times (1-p)$. These values are equal to 12.83 22.81 22.81 and 40.55 respectively. Using the function *CHISQSTAT*:

```

      V R+E CHISQSTAT 0
[1]   R+/, ((O-E)*2)÷E
      V

      P+.36
      E+2 2p99*(P*P), (P*1-P), (P*1-P), (1-P)*1-P
      E CHISQSTAT TRANCOUNT STATES
21.1

```

As always with Statistics, solving one problem only leads to another. The value 21.1 measures how far the observed transition frequencies differ from what we would expect if the data had been generated by coin tossing. The bigger that value, the more the frequencies deviate from those expected. But what is the critical value such that values above would allow us to reject safely the coin-tossing model? The answer to this question either comes from a lot of quite sophisticated mathematics that result in tables of the chi-square distribution, or a modern alternative is to use simulation. We use the latter here as it illustrates further methods of using APL and also because being less technical, it may help readers who are baffled by statistical tables and the 'magic' that surrounds them!

What is required of the simulation is to simulate coin-tossing, producing repeatedly, 100 tosses of a coin with probability of heads equal to 0.36, calculating for each set of 100 tosses, the 2 by 2 frequency transition matrix and its chi-square statistic. We then compare our observed value of 21.1 with the set of simulated values. We are then able to judge how atypical the value of 21.1 is.

The program to do this is called *SIMULATION*. The hard work of the program is already done for us - all we need is a loop to repeat the simulation of 100 tosses the required number of times. The actual simulation can be achieved by constructing a vector *S* of 36 1's and 64 2's, and then indexing this vector using $?100\rho 100$ (see my article on simulation in Education Vector April 1989).

```

V R+SIMULATION N;C;P;E;S
[1] R+N\rho 0 \rho C+1 \rho P+0.36 \rho P=probability of heads
[2] \rho Calculate expected frequencies given p
[3] E+ 2 2 \rho 99*(P*P),(P*1-P),(P*1-P),(1-P)*1-P
[4] \rho Calculate a vector of 36 1's and 64 2's
[5] S+(36\rho 1),64\rho 2
[6] \rho Now loop, assigning each answer into R
[7] lp:R[C]+E CHISQSTAT TRANCOUNT S[?100\rho 100]
[8] \rho +(N>C+C+1)/lp
V

```

If you have a reasonably fast machine, then it is not too time-consuming to do say, 100 simulations and store the result ready for display using a histogram. Even a small number of values gives you some idea as to the expected value of the chi-square statistic (which is about 2). Here are 10 typical results.

```
5 2\r SIMULATION 10
```

(The expression '5 2\r' formats the results as real numbers using 5 characters with 2 decimal places)

```
5.41 .06 .59 7.84 2.37 3.20 1.17 2.62 1.43 1.85
```

If you have patience with your machine to do 100 simulations and plot the results as a histogram then your results should look something like mine:

<i>Two-step transitions</i>	<i>Probabilities</i>	<i>Two-step Transition Probs</i>
1→1+1	.2×.2	\ -----(.2×.2)+(.8×.6)
1→2+1	.8×.6	/
1→1+2	.2×.8	\ -----(.2×.8)×(.8×.4)
1→2+2	.8×.4	/
2→1+1	.6×.2	\ -----(.6×.2)×(.4×.6)
2→2+1	.4×.6	/
2→1+2	.6×.8	\ -----(.6×.8)×(.4×.4)
2→2+2	.4×.4	/

The computations in the right-hand column are precisely the four calculations done in multiplying P by itself, hence the two-step transition probabilities are given by $P+ . \times P$. Similarly, the three- step transition matrix is given by P raised to the power 3 or $P+ . \times P+ . \times P$.

It is very interesting to see what happens when you raise P to successive powers. This is easily done as indicated above or as follows:

$$\begin{aligned}
 &P^2+P+ . \times P \\
 &P^4+P^2+ . \times P^2 \\
 &P^8+P^4+ . \times P^4 \\
 &P^{16}+P^8+ . \times P^8
 \end{aligned}$$

$$\begin{aligned}
 &P^2 \\
 &0.52 \quad 0.48 \\
 &0.36 \quad 0.64
 \end{aligned}$$

$$\begin{aligned}
 &P^4 \\
 &0.4432 \quad 0.5568 \\
 &0.4176 \quad 0.5824
 \end{aligned}$$

$$\begin{aligned}
 &P^8 \\
 &0.42894592 \quad 0.57105408 \\
 &0.42829056 \quad 0.57170944
 \end{aligned}$$

$$\begin{aligned}
 &P^{16} \\
 &0.428571674 \quad 0.571428326 \\
 &0.4285712445 \quad 0.5714287555
 \end{aligned}$$

As you can see, the rows of powers of P rapidly become equal as the power increases. This is of course means that the probability of jumping to the states 1 or 2 in a large number of jumps depends very little on the starting state. This loss of memory, shows that not all the properties of coin tossing have been lost in proposing a more complicated model - there is still a long-run chance of state 1 occurring, given in this case by .42857 or more correctly $.6 \div (.6 + .8)$. Of course the long-run chance of state 2 is 1 minus this or $.8 \div (.6 + .8)$.

It turns out that these long-run probabilities as a vector can be calculated very easily in APL for transition matrices P . (Basically a transition matrix is a square matrix, each of whose elements are non-negative and whose row-sums are 1.) I offer the function here with no explanation whatsoever and as a one-liner. (It comes from a paper by myself and Alan Hawkes recently submitted to Transactions in Reliability IEEE). I hesitate to guess how many lines of Fortran code would be necessary to do this!

```

▽ R←LONGRUNPROP P
[1] R←((Tρ0),1)⊖(⊖P-(ρP)ρ1,(T+1+ρP)ρ0),[1]1
▽

```

Applying this function to the probability transition matrix P of a 3 state Markov chain given by:

```

0.19 0.75 0.05
0.17 0.18 0.65
0.13 0.11 0.76

```

```

5 3▽LONGRUNPROP P
0.148 0.221 0.631

```

So we are able to deduce that for example, the long-run probability of state 3 occurring is .63.

By now you are perhaps wondering (or more likely have given up wondering) just what all this has to do with the weather. Needless to say, the model that I have illustrated has many applications in real-life. The meteorological application I had in mind was the pattern of successive days where 1 indicates rain occurs, and 2 indicates rain did not occur. In real life, this is an oversimplification but nevertheless is quite successful in modelling the occurrence or non-occurrence of rain at a particular place, at least for a relatively short time period. Of course there's always the possibility of snow! And on that note, I bring this article to a rapid close.

I-APL News - December 1989

by Anthony Camacho

News of the Apple Port

I hear from Rolf Levenbach that he was able to load and save a workspace on the Apple for the first time on 12 December 1989! He also has a cursor (which I had not even realised was a problem). Now we have to produce documentation and do Beta testing. While that is going on we may be able to produce some workspaces to give access to the Apple machine facilities. Won't it be marvellous to be able to release another port.

A Traditional APL Keyboard for I-APL

F. D. H. van Batenburg of the Dutch Group has been working to produce the traditional keyboard - as for the golfball typewriter, and has done everything except the ALT key combinations. Paul is finding out where in the code the ALT key is captured and then we will be able to offer a patch which will enable professional APLers to demonstrate I-APL without getting lost looking for backslash-bar.

Large Workspaces

We have been asked to provide access to more memory on machines such as the Archimedes and PC clones which usually have half a megabyte or more. There doesn't seem much sense in doing this without removing limitations which were also part of the drive to save space in 8-bit machines such as BBC and Apple. For example we want transcendentals accurate to more than seven places and better floating point precision.

Paul Chapman and Dave Ziemann are therefore looking into the question to decide which is the best way to enhance I-APL to provide a 32-bit version. What enhancements do you think we should include? The candidates so far are: LEV and DEX, lower case names, the RANK operator, long direct definition lines, increased accuracy/precision (either longer BCD or IEEE), keyboard enhancements, FSCREEN access to attributes and several technical improvements which should result in shorter or faster code.

Order Form: I-APL Ltd February 1989

The address for queries is:
 2 Blenheim Road
 St Albans
 Herts AL1 4NR

The address for orders is:
 56 The Crescent, Milton
 Weston-super-Mare
 Avon BS22 8DU

Please supply:

Part	Item	(pounds sterling)		
		No copies	Price	Total
PC12	PC & clones disk & manual	_____	£4.50	_____
BBC12	Master 80T disk & manual (also B with 32K sideways RAM)	_____	£4.50	_____
A12	Archimedes disk & manual	_____	£4.50	_____
M2	Manual (PC/BBC/ARM/Nimbus)	_____	£2.50	_____
T3	Tutorial by Thomson & Alvord	_____	£2.50	_____
E4	Encyclopedia by Helzer	_____	£5.50	_____
S5	APL in Social Studies by Traberman	_____	£2.50	_____
N6	APL Programs for the Maths Classroom by Norman Thomson (pub Springer Verlag)	_____	£14.50	_____

Packing charge per order _____ £1.00
 Prices include VAT (reg no 479 1634 13) and postage

I enclose cheque to I-APL Ltd for: _____

N.B. We take no credit cards.

Name: _____

Full address: _____

Please send me a VAT invoice (tick) Vat is 27p on disk only.

Office use only		
-----------------	--	--

APL90: Accepted Papers

Received on January 16th 1990

Author(s)	Short Title of Paper
Jan Ahlqvist	SRS Service System
Manuel Alfonseca	Neural Networks in APL
Tony O'Hagan Jake Ansell Alan Sykes David Eastwood John Searle	The APL Statistics Library Project
Heikki Afiola Pirkka Peltola	Integrating APL with Symbol Manipulation, Numerical Software and Graphics
J Philip Benkard	Nonce Functions
Robert Bernecky	Acorn APL to C on Real Numbers
P Bottoni P Mussio M Protti	Definition of Image Interpretation Strategies in APL
Renato Capra	Preliminary Mesh Checking for Structural Analysis
Edward Cherlin	APL Trivia
Wai-Mee Ching	Automatic Parallelization of APL-style Programs
Yap Siong Chua	Image Processing Algorithms Facilitated by APL2
Don Erickson	Technical Support Program for APL-related Questions
Frank Evans Jan Jantzen	A Structured Approach to Analysis and Design of Complex Systems
William G Foote	APL2 Analysis and Design of Mortgage-backed Securities
Garth H Foster Abdelatif Elguori Franklin Liu	An Integrated Programming Environment for APL2/PC
Ralph L Fox	Color APL Beautiful
Erik S Friis	APL2 and MIDI
Morten Kromberg Martin Gfeller	An Application Development Platform

Author(s)	Short Title of Paper
Jean Jacques Girardot	The A+ Programming Language, a Different APL
Jean Jacques Girardot	Arrays and References
Janice H Cook Leo H Groner	QMOD, an Analytic Response-time Model for Distributed Systems
Alan G Hawkes	Markov Processes in APL
Ferdinand Hendriks Wai-Mee Ching	Sparse Matrix Technology Tools in APL
Roger Hui Kenneth E Iverson Arthur T Whitney	APL\?
Kenneth E Iverson	An APL Hornbook
Torben Iverson	APL for Economic and Management Control in KTAS
Eeva-Liisa Kaski	End-user KESSU
Andrew V Kondrashev	The Family of Soviet APL Systems
Gerard A Langlet	The Travelling Salesman Problem, Revisited with APL
Timo Laurmaa	Desktop Publishing on Mainframe with APL and Ventura Publisher
Edward Y H Lin Dennis L Bricker	Implementing the Recursive APL Code for Dynamic Programming
Guy Barker Douglas J Keenen Herman van Loon	Conscientious Programming using PMA
Jim Lucas	Programming Ecology
Sven Gunnar Lann	DEMOS - a PC-system for Population Projections for Small Areas
Hans-Peter Meinzer	Visualization of Medical Tomography Image Series
Ole M Meyer	An Insurance Simulation Model
Alexey I Miroshnikov	Algorithm Alterable Models and APL
Trenchard More, Jnr	Looking Beyond APL2 and NIAL
Thomas M Olsen	Multi-axis NC Postprocessor for Machining Centers
Panagiotis Pantziarka	Object Oriented Database using Frames in Second Generation APL

Author(s)	Short Title of Paper
Raymond P Polivka	Reading to Write
Thomas J Pritchard	IBM System/370 Channel Programming using APL
Steven I Promisel James Merrill	Managing a Diamond Jewelry Manufacturing Business using APL
Ursula Recker Michael Rys	A Preferable Look - APL in Window-based Environments
Jack Rudd	Toward a Common Prototyping Language
William A Rutiser	Object Oriented Programming of X Window System
Jorgen Sauer mann	A Parallel APL Machine
Charles A Schultz	Writing Applications
D Smellie F Evans	Structured Expert System Design
Adrian Smith	Designing an APL Typeface using a Partial Implementation of PostScript in APL
Howard J Smith Jnr	What's Ahead for 2000AD
Howard J Smith	Some Uses of Truncated Boolean Vectors in Analysis
Donald Soule	Stability in a Sea of Change
Andreas Geyer-Schulz Johann Mitlohner Alfred Taudes	An APL-Simulator of non-Von Neumann Computer Architectures
Timo Teileri Toivo Olkkola	LYYTI - Integrated Design and Control System
Norman Thomson	APLEGANCE - The Art of Staying Within One's Depth
Tom Springall Gustav Tollet	A Shading Approach to non-convex Clipping
David Weintraub	APL2OS: Design Considerations for a Nested Array File System
James G Wheeler	Compiled Language Interface in APL*PLUS II
Charles N Winton	6749: APL's Deal for the Florida Lottery
Jonny Osterman	Very High Quality User-Interfaces and Fast Data Filing Using a PC.



Renaissance Data Systems
Enlightenment Thru Information Processing
ALL APL BOOKS IN PRINT
 Catalog Excerpts June 1, 1989



If we do not carry it (in English, at least), we dare you to find it elsewhere!
 Specializing in books on:
APL AS A TOOL OF THOUGHT!

ACCOUNTING STRUCTURED IN APL , Ijird. 1984, 491p.	\$10.00
The basic principles of accounting with numerous APL expressions to model them.	
ABSTRACT ALGEBRA - A COMPUTATIONAL APPROACH , Simms. 1984, 491p.	\$55.95
An introductory course in the basic concepts of algebra and fundamental algebraic algorithms. A methodology for investigating computational questions. Includes lemmas, theorems, and corollaries.	
COMPUTING IN STATISTICAL SCIENCE THRU APL , Anscombe. 1981, 426p.	\$39.00
Lots of statistics and APL. His workspaces are available from Yale University. A classic.	
MATHEMATICAL EXPERIMENTS ON THE COMPUTER , Grenander. 1982, 525p.	\$59.50
Case studies using APL in a number of fields, including statistics, linear algebra, geometry, asymptotics, neural networks, invariant curves. Detailed description and analysis of APL functions in all topics discussed.	
PROBABILITY IN APL , Alvord. 1984, 142p.	\$19.50
A delightful, yet serious development of 31 functions for fun and learning with roll, deal, binomial distributions, combinations, permutations, geometric distributions, and the World Series. Friendly examples. No previous APL required.	
Diskette for above containing functions covered in book.	\$22.00
APL - THE LANGUAGE AND ITS ACTUARIAL APPLICATIONS , de Kerf, Goovaerts, Stiers. 1987, 223p.	\$89.00
Introduction to APL. Loss reserves, credibility, probability, numerical analysis, forecasting, with APL functions.	
COMPUTER GRAPHICS FOR DESIGNERS AND ARTISTS , Kerlow and Rosebush. 1986, 298p.	\$44.95
A well organized, beautifully done reference book. Step-by-step explanations of terms and techniques from bits and bytes to 3-dimensional imaging. Many explanatory illustrations, most in color. No APL, but, says Rosebush, APL thinking and data structures are implicit throughout.	
APL2 AT A GLANCE , Brown, Pakin, Polyvka. 1988, 444 p.	\$29.00
Solid, unimimidating, introduction to APL2. Clear illustrations, modern exercises in each chapter. Gets you started.	
THE APL IDIOM LIST , Perlis and Rugaber. 1977, 50p.	\$ 6.50
Funded by Mobil. Written at Yale. Very terse, but includes nuggets of gold.	
APL - ADVANCED TECHNIQUES AND UTILITIES , Berquist. 1986, 450p.	\$44.95
Good discussion of alternative approaches to a wide range of programming tasks - efficiency, searching, dates, workspace documentation, file design, boolean techniques, financial utilities. Many idioms.	
Diskette for above.	\$15.00
Contains all functions described in text. Formatted for APL*PLUS/PC, uploadable to Sharp APL, VS/APL, APL2.	
APL AS A TOOL OF THOUGHT, PROFESSIONAL DEVELOPMENT SEMINARS, NY/SIGAPL	
They cover a wide range of topics in business and education. 1987 - logic, insurance, statistics, A. I., accounting, fractals, linked lists. Other years: Teachers toolbox, computer science, biology, graphics, engineering, data bases, and more.	
1984, 1985, 1986, 1987. 120 to 150 pages each (1983, 47p. \$10.00)	each \$15.00
All five as a package.	\$55.00

Name: _____	Date: _____	Please make check or money order payable in \$ U. S. and send with this form to: Renaissance Data Systems P. O. Box 20023 Park West Finance Station New York, New York 10025-1510 U.S.A. (212)864-3078
Street: _____		
City: _____	Prov./State: _____	Postal Code: _____
Country: _____	Telephone Day: _____	Evening: _____
Title _____	Price _____	Quantity _____ Total _____
Title _____	Price _____	Quantity _____ Total _____
Title _____	Price _____	Quantity _____ Total _____
Subtotal: _____		
Postage and handling - U.S. and Canada (\$2.50 + \$2.50 each item over \$25.00): _____		
- International (surface mail add 15%, airmail add 50%): _____		
NYC Residents please add 8 1/4% sales tax, NY State, 4%: _____		
Please send complete CATALOG of 64 Titles: <input type="checkbox"/>		Total: _____

APL Product Guide

Compiled by Alison Chatterton

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 do depend on the alacrity of suppliers to keep us informed about their products so that we can update the Guide for each issue of VECTOR. Any suppliers who are not included in the Guide should contact me to get their free entry - see address below.

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. Where no UK distributor has yet been appointed, the vendor should indicate whether this is imminent or whether approaches for representation by existing companies are welcomed.

For convenience to readers, the product list has been divided into the following groups:

- * Complete APL Systems (Hardware & Software)
- * APL Timesharing Services
- * APL Interpreters
- * APL Visual Display Units
- * APL character set printers
- * APL-based packages
- * APL Consultancy
- * APL Training Courses
- * Other services
- * Vendor addresses

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.

Note: 'poa' indicates 'price on application'.

All contributions to the APL Product Guide should be sent to Alison Chatterton, at the address on the inside back cover.

COMPLETE APL SYSTEMS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Dyadic	APL Coprocessor	3,500+	32-bit coprocessor board for IBM PC. NS32000 cpu with FPP, up to 4Mb RAM, 16Mb virtual memory. Software includes Unix V.2, Dyalog APL, graphics support, DOS interface. Provides multi-user Unix/DOS environment.
	IBM 6150	15,000+	Multi-user Dyalog APL system with Fast 32-bit RISC processor, FPP, up to 8Mb RAM, 210Mb Disk, 16 users. Interface to SQL, graphics and APL support for standard IBM peripherals.
	Altos 3068	25,000+	Multi-user Dyalog APL system with MC68020 cpu & MC68881 FPP. Also features a LAN which supports IBM PCs as Dyalog APL terminals.
	Sun 3	15,000+	Multi-user Dyalog APL systems which can be configured as a network of workstations and/or a traditional time-sharing cpu. With its 25MHZ 68020 cpu, the Sun 3/200 is the fastest APL microcomputer on the market.
M.B.T.	MBT Series 10	poa	UNIX/68010 based multi-user APL system
	TORCH	poa	68000/Z80 multiprocessor
MicroAPL	Aurora	20,000+	Multi-user APL computer using 68020 CPU. Std. configuration 2Mb RAM, 16 RS232 ports, 68 Mb hard disc, 720K diskette
	Spectrum	7,000+	Expandable multi-user APL computer using Motorola 68000. Std. configuration 1 Mb RAM, 12/36 Mb disc, 12 ports.
	Atari 1040ST	799	1 Mb Mono/Colour System, includes 1 Mb disc drive & mains transformer built into Console.

APL TIMESHARING SERVICES

COMPANY	PRODUCT	PRICES(£)	DETAILS
Mercia	APL*PLUS	poa	STSC's Mainframe Service - MAILBOX etc.
I.P. Sharp	SHARP APL	poa	International Network application systems and public databases.

APL INTERPRETERS

COMPANY	PRODUCT	PRICES(£)	DETAILS
APL Software	Dyalog APL	1,000-8,000	See Dyadic Systems entry
Cocking/Drury	APL*PLUS/PC Rel 9	535	STSC's full featured APL for IBM PC, PC/AT and PS/2. Upgrades are available from V.* and earlier releases.
	Run-time	poa	Closed version of APL*PLUS/PC which prevents user exposure to APL.
	APL*PLUS II	2,000	includes 1st year maintenance
	runtime	250	
	site licence	poa	
	APL*PLUS UNIX	poa	STSC's 2nd gen APL for all major Unix computers and workstations.
	APL*PLUS VMS	poa	STSC's 2nd generation APL for DEC VAX computers running under VMS.
	APL*PLUS mainframe	poa	Enhances VS APL with many high productivity features. For VM/CMS, MVS/TSO: offers simple upgrade from VS APL.
Dyadic	Dyalog APL	795-10,000	2nd gen. APL for UNIX systems, e.g. IBM 6150, Sun, Vax, NCR, HP9000, AT&T, Altos, Apollo, Whitechapel, Sperry, etc.
Gen. Software	APL*MYRIADE	poa	Runs on Texas Instruments TI990 range.
IBM UK	IBM PC APL2	348	APL2 for the IBM PC. Program 5799-PGG/1, PRPQ number RJ-0411. From all IBM dealers.

	Mainframe APL2	poa	APL2 release 3
	APL2 Appl. Envt	poa	
Inner Product	VIZ::APL	250-350	8-bit Zilog Z-80 CP/M
	APL*PLUS/PC	600	See under Cocking & Drury
M.B.T.	Dyalog APL	poa	See Dyalog Systems entry
	MBTAPL	poa	Enhanced Dyalog APL for MBT hardware.
	VIZ::APL	poa	Customized for TORCH hardware
MicroAPL	APL.68000	1,000-2,000	Full implementation with component files, error trapping etc. for SPECTRUM, HP300, SUN, NCR etc.
	QL/APL (keyword)	87	Full keyword APL for QL with many extra features.
	QL/APL (APL chars)	87	VSAPL compatible APL for QL with many extra features.
	APL.68000 Apple Macintosh	87	
	APL.68000 Apple Mac II	260	
	APL.68000 Amiga	87	Full APL interpreters with support for windows, mouse, graphics etc.
	APL.68000 for Atari ST	87	
	APL*PLUS/PC - REL B	450	
	APL*PLUS II	1,395	
Portable	PortAPL	\$195	IBM PC Software
		\$275	Macintosh
		\$2,995	DEC VAX
I.P. Sharp	APL*PLUS/PC	480	STSC APL
	Upgrade V7 to V8	100	
	Upgrade any to V8	200	
	Sharp APL/PCX V1	1,650	For IBM XT/AT 370
	Sharp APL/PC V2	40	shareware for IBM PC/XT
	Sharp APL	poa	for IBM mainframes
Uniware	APL*PLUS/PC	call	STSC's full feature APL for IBM PC/XT/AT and PS/2
	Run-Time	call	Closed version of APL*PLUS/PC which prevents user exposure to APL
	APL*PLUS/UNIX	call	STSC's full feature APL for UNIX based computers.
	APL*PLUS II	call	STSC's full feature APL for 386 computers

APL VISUAL DISPLAY UNITS

COMPANY	PRODUCT	PRICES (£)	DETAILS
Dyalog	Lynwood j300	1,560	Monochrome ANSI 3.64 APL vdu, 15-inch high quality screen, Tek graphics, local macro keys.
	Lynwood j500	2,295	Colour ANSI 3.64 APL vdu, 15-inch high quality screen, Tek graphics, local macro keys.
	IBM 3163	791	Low-cost Monochrome APL vdu. Supports downloaded Dyalog APL font.
	IBM 3164	1,093	Low-cost Colour APL vdu. Supports downloaded Dyalog APL font.
Farnell	Tandberg TDV 2221	995	Ergonomic design APL terminal, 50-19200 baud, 1518if5 anti-reex screen, low profile keyboard
	Tandberg TDV 2271	1,195	Combined APL/ANSI ergonomic terminal as above.
Gen. Software	Mellordata	400	Second-hand Elite 3045A

M.B.T.	various		Contact MBT for details
Meta Technics	IBM EGA compatible	299	Emulates EGA & Hercules, Half Card
MicroAPL	Insight VDT-1	795	Inexpensive APL VDU
	Insight GDT-1	1,450	With monochrome graphics
	Concept201	1,295	APL VDU with 8 page memory
	Concept 201G	1,850	Graphics VDU
Shandell	HDS2010	1,215	ANSI3.64 DEC VT52/100/220 compatible. 15" tilt/swivel screen, low profile keyboard 8 page memory, windows, viewports, 80/132 columns, full overstrike, 2 or 3 comms. ports, 55 PF keys, NVM storage.
	HDS2010G/GX	1495+	As above plus Tektronix 4014, Retrographics VT640/D0640 and Visual 500 compatible. 1024 x 390 or 1024 x 780 resolution.
Tektronix	4114B	13,500+	19" D.V.S.T.:Graphics: 3120 x 4096 displayable; Intelligent: up to 800K memory; APL keyboard (option 4E)
	4125	21,550+	19" 2D colour graphics; Workstation (1280 x 1024);Intelligent: up to 800K memory; APL keyboard (mod AP)
	4128	26,822+	As 4125 plus 3D wireframe

APL PRINTERS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Datatrade	Datasouth DS180+	1,295	180 cps matrix printer with 4K buffer, 9 x 7 dot matrix and APL option.
	Datasouth DS220	1,695	Letter quality; graphics capability, APL option (both available with IBM Twinex or Coax interface).
Dyadic	IBM 4201 Proprinter	poa	100, 200, 40(niq) cps, matrix printer, with graphics. Supports downloaded Dyalog APL font.
	Toshiba P351	poa	24 pin high-quality matrix printer 100 cps letter quality, 192 cps draft.
Inner Product	Epson FX80	500	Soft char. set, 180 cps, 80 column
	Anadex 9620	1,150	200 cps., 132 col., tractor feed
	Siemens PT88	620	180 cps., 80 col., silent
	TGC Starwriter	1,180	40 cps., letter quality
M.B.T.	Facit 4565	poa	40 cps letter-quality
	Facit 4510/11/12	poa	Matrix printers
MetaTechnics	Quen-data	295	Low-cost APL Daisy-wheel printer
MicroAPL	Datasouth DS180+	1,195	See Datatrade entry
	Philips GP480	2,137	Matrix printer with letter & draft quality and APL (480 cps).
	Qume Letterpro20	549	APL/ASCII Daisy-wheel printer

APL PACKAGES

COMPANY	PRODUCT	PRICES(£)	DETAILS
APL-385	APL-385	50(PC), 125(mf)	including ...
	FSM-385		Screen development
	DRAW-385		Screen design
	DB-385		Relational W.S.
	GEN-385		Miscellaneous Utilities
APL Software Ltd (mainframe)	RDS	poa	Relation Data Base System

	PANEL	poa	Fullscreen management system
	PFS	poa	Program File System - APL Systems development aid
	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
Beta-plan	BETA-FONT	poa	Multiple font PC character generator. Dealers required for non-Scandinavian countries.
APL IMPETUS	Impetus	poa	Hierarchical Planning System
Butel	Merlin	5,000	Mainframe APL spreadsheet runs under VM/CMS, TSO, VSPC
	Merlin/PC	poa	Version for APL*PLUS/PC
Cocking/Drury (for VSAPL)	E'EMENTS & SHAREFILE	poa	Component files, quad- functions & nested arrays for IBM VSAPL under VM/CMS & MVS/TSO
	COMPILER	poa	The First APL compiler.
	FILEPRINT	poa	Print APL component files
	FILECONVERT	poa	Convert non-APL files to APL files
	FILEMANAGER	poa	Extends APL primitives to database management
	TOOLS + UTILITIES	poa	APL Software development tools
	DATAPORT	poa	Information Centre spreadsheet incorporating data exchange between APL and FOCUS, IFPS, SAS, APL/DI, ADRSII, LOTUS123, MULTIPLAN, DIF files
(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.
	FMT	poa	Full featured FMT for APL2
	WSDOC	poa	Workspace documentation utilities
	FILEMANAGER	poa	Extends APL primitives to database management
(microcomputer)	APL*PLUS PC Tools	325	Utilities including: RAM disk, full screen data entry, menu input, report generation, exception handling and games.
	IRMA Module	120	327x IRMA support.
	Fin & Stat. Library	350	Financial & statistical routines
	SPREADSHEET MGR	195	APL-based spreadsheet for APL*PLUS/PC. Cell arithmetic; transfers to ASCII, LOTUS
E&S	PROTOPAK		Packages for prototyping management information systems - consisting of: PC & mainframe modules
	RMS		Relational databases.
	AMS		Multi-dimensional arrays
	RAMS		Combined RMS & AMS.
	BMS		Dynamic financial modelling & forecasting
	FMS		Full-screen handler for APL*PLUS/PC. (AP124-based)
	CMS		Communications package.
	SOS	poa	Scheduled ordering and stock control.
FASTCODE	FASTCODE, MONITOR	\$299	Assembler written monitor for APL applications in APL*PLUS/PC
Gen. Software	PROPS	500+	Spreadsheet system for Product and/or Project Planning.
H.M.W.	INPUT	poa	Matrix manipulation package for data entry & report generation

	POWERTOOLS	poa	For faster primitive screen handling, forms validation (SAA / CUA conforming). Includes support for mice, touch-pads etc.
	PRINTPAK	poa	Block printing for VM/CMS
	VIEWPAK	poa	AP124 Protocol emulator for IBM/PC
	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	PC Utilities including: APLMAC (windows); Unlock (unlocks functions in .AWS); DTEX (text and spreadsheet imp/exp). Mostly available in English or French.
Inner Product	Viewcom	150	Control Viewdata from APL
	APL/dBASE II	150	Interface APL with dBase II
	APL/dBASE III	150	Interface APL with dBASE III
	APL/LOTUS	150	Interface APL with Lotus
	APL/WORDSTAR	150	Interface APL with Wordstar
	APL/MULTIPLAN	150	Interface APL with spreadsheet
	CEMAS	3,500	EEC monetary and agrimonetary analysis.
Interprocess	IEDIT	1900-3200	Full screen APL2 editor with immediate APL execution, and a full-screen debugger
(mainframe)	AFM	8200-9800	High performance component and keyed file system (VS APL and APL2)
	Format	1650	A QuadFMT data formatter for VS APL and APL2
	FSM124	1650	AP124 programming for APL applications without GDDM (APL2)
	PowerCode	1300	External functions for APL2
	CALL/AP	3000	for calling non-APL programs (VS APL and APL2)
	UCF	1800	Inter-user data transfer for VM users via IUVC
(PC)	AFM	115	Single user component and keyed files for APL2/PC
M.B.T.	RHOMBUS	poa	Integrated Otsu3suce System
	HASLEMERE	poa	Hotel Accounting System
Mercia	STATGRAPHICS 3.0	685	Integrated stat. graphic system for PCs.
	Upgrade 2.1 to 3.0	395	
	Upgrade 2.6 to 3.0	175	
	MICROSPAN	250	Comprehensive APL tutor
	LOGOL	poa	Logistics management system for PC, Forecasting, Inventory Control, Scheduling, Distribution, etc
MetaTechnics	MetaScreen	99	Full-screen handler for APL*PLUS/PC, based on VSAPL AP124
	MetaPack	495	Comprehensive utilities package for APL*PLUS/PC. Includes: MetaScreen, MetaWS, Browse, Toolbox, Numeric Editor.
	APL-IEEE488	99	Controls IEEE488/GPIB Bus from APL*PLUS/PC.
	PLOT/PC	99	2D & 3D Graphics package. Includes interactive diagram Editors.
	Browse	99	Scrolling of DOS files, large APL variables.
	ADAPTA DLS	poa	Production & purchasing scheduling for process manufacturing.
	ADAPTA MSP	poa	Job-shop loading & scheduling for multi-stage production.
MicroAPL	MicroTASK	250	Product development aids
	MicroFILE	250	File utilities and database
	MicroPLOT	250	Graphics for HP plotters etc

	MicroLINK	250	General device communications
	MicroEDIT	250	Full screen APL editor
	MicroFORM	250	Full screen forms design
	MicroSPAN	250	Comprehensive APL tutor
	MicroGRID	poa	Ethernet & other networking
	APLCALC	400	APL spreadsheet system
	MicroPLOT/PC	250	For APL*PLUS/PC product
	MicroSPAN/PC	250	For APL*PLUS/PC product
	PC TOOLS Vol 1	295	
	STATGRAPHICS Rel 1	495	
	STATGRAPHICS Rel 2	535	
I.P. Sharp	GLOBAL LIMITS	poa	Exposure management for banks
	IPSA/CONNECT	poa	Mainframe to micro link
	MAILBOX	poa	Electronic Mail
	MAILBOX/PC V.2	65	Full screen front end to IPSA mailbox
	upgrade to V.2	15	
	NEWSFLASH	poa	Real time message exchange
	VIEWPOINT	poa	4GL - Info centre product
Sugar Mill	Stat 1	\$129.95	Statistical toolbox, menu driven
Uniware (mainframe)	STSC's ENHANCEMENTS	10,715	Quad-functions & nested arrays for IBM VSAPL under VM/CMS and MVS/TSO
	STSC's SHAREFILE	10,715	Component files for IBM VSAPL under VM/CMS and MVS/TSO and for IBM APL2
	PROGRAMMER TOOLS & UTILITIES	5,715	
	FILEPRINT	5,715	
	FILESORT	5,715	
	FILECONVERT	5,715	
	FILEMANAGER (EMMA)	5,715	STSC's database package.
	APL*PLUS COMPILER	21,430	First APL compiler. Complements APL*PLUS enhancements and Sharefile under VM/CMS and MVS/TSO.
	EXECUCALC	3,995	Mainframe spreadsheet compatible with VISICALC and part of LOTUS 1-2-3 under VSAPL (VM or TSO).
(microcomputer)	STATGRAPHICS	call	Statistics & Graphics for PCs.
	STATGRAPHICS FCA	call	An add-on module to STATGRAPHICS: Factorial Correspondence Analysis.
	APL*PLUS/PC TOOLS - VOL1	call	Incl. 327 x IRMA support, RAM disk, full screen data entry, menu input, report generation, games.
	- VOL2	call	Incl. File documentor, screen editor, exception handling.
	SPREADSHEET MNGR	call	APL spreadsheet with built-in ASCII, LOTUS and SYMPHONY interfaces.
	APL*PLUS/PC FIN. & STAT.LIBRARY	call	Collection of financial and statistical utilities.
	POCKET APL	call	Smaller version of APL*PLUS/PC.
	UNIASM	call	Collection of assembler routines for APL*PLUS/PC users.

	UNITAB	call	APL*PLUS/PC spreadsheet-like data entry and validation system.
	The APL DEBUGGER	call	First released APL*PLUS/PC debugger.
	OVERLAYS	call	Fast assembler routines to handle overlays in APL*PLUS/PC.
	R:BRIDGE	call	Interface between APL*PLUS/PC & R:BASE 5000.
	DMA	call	A version of EMMA (APL database manager) for APL*PLUS/PC users.
	APL2C	call	Interface between APL*PLUS/PC and DATALIGHT C language.
	ADAPTA/DLS	call	Production & purchasing scheduling for process manufacturing.
	PLOT*PLUS	call	Graphics package for APL*PLUS/PC
Warwick University	BATS	500	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.

APL CONSULTANCY

(prices quoted are per day unless otherwise marked)

COMPANY	PRODUCT	PRICES(£)	DETAILS
APL Consulting	Consultancy	180-550	Expertise in APL system design, project management, prototyping, financial applications, decision support systems, MIS, links to non-APL systems, documentation.
APL People	Consultancy	from 120	Consultants available at all levels, with experience in: VS APL, APL*PLUS, APL2, Sharp APL, Dyalog APL, APL68000, C/Unix, TSO/MVS, VM/CMS, graphics. Additional skills available: OR, statistics, financial, commerce, retail, engineering.
APL Software Technology	Consultancy	poa	Technical & business systems, micros, networking & communications a speciality.
Buckland Management Systems	Consultancy	poa	Business and Technical systems in commerce and industry. Designing, programming and implementing applications of O.R and statistics.
Camacho	Consultancy	poa	Specialising in programming & manual writing.
Chapman	Consultancy	150-300	24-hour programmer: APL, C, assembler, graphics; PC, mini, mainframe, network.
Cocking/Drury	Consultancy	175-275 275-350 300-450 400-600 450-750	Junior consultant Consultant Senior consultant Principal Consultant Managing consultant
Peter Cynrax	Consultancy	100-150 120-200 150-300	Junior Consultant Consultant Senior Consultant
Delphi	Consultancy	poa	Specialising in management reporting systems and APL on microcomputers.
Dyadic	Consultancy	poa	APL system design, consultancy, programming & training for Dyalog APL, VSAPL, APL*PLUS, IPSA APL etc.
E & S	Consultancy	150-250	System prototyping: all types of information system.
FASTCODE	Consultancy	poa	Specialise in improving performance of APL applications on micros & mainframes.
Gen. Software	Consultancy	100+	

H.M.W.	Consultancy	poa	System design consultancy, programming. HMW specialize in banking and prototyping work.
Ian A. Clark	Consultancy	poa	Computer-based Information Systems implementation where acceptance is critical. APL on PC and Macintosh. Human Factors of HCI; novice ease of use; online assistance; training courses; distance-learning materials.
Inner Product	Consultancy	200	On-site micro-mainframe APL, PC/DOS & Assembler
Lloyd Savage	Consultancy	poa	Decision support, particularly specialising in Sales & Marketing systems.
M.B.T.	Consultancy	poa	
Mercia	Consultancy	poa	APL*PLUS & VSAPL consultancy.
MicroAPL	Consultancy	poa	Technical & applications consultancy.
M.T.I.	Consultancy	poa	Specialise in Maintenance and development of existing APL systems
Parallax	Consultancy	\$750	Introductory APL, APL for End-user & Advanced Topics in APL
QB On-Line	Consultancy	250	Specialising in Banking, Financial & Planning Systems.
Rochester Group	Consultancy	poa	Specialise in MIS using Sharp APL
I.P. Sharp	Consultancy	poa	Consultancy & support service world-wide.
Uniware	Consultancy	call	Junior to managing consultancy in APL

OTHER PRODUCTS

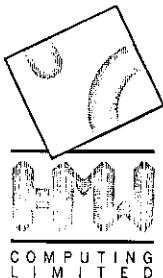
COMPANY	PRODUCT	PRICES(£)	DETAILS
APL People	Employment Agency	poa	Permanent employees placed at all levels. Contractors supplied for short/long-term contracts, supervised or unsupervised. Executive Search service available.
HMW	Employment	poa	Contractors and permanent employees placed.
I.P. Sharp	Productivity Tools	poa	Utilities for systems, operations, administration & analysts; auxiliary processors, comms software, international network.
	Databases	poa	Financial, aviation, energy and socioeconomic.

VENDOR ADDRESSES

COMPANY	CONTACT	ADDRESS & TELEPHONE No.
APL 385	Adrian Smith	Brook House, Gilling East, York. Tel: 04393-385
APL Consulting	Jill Moss	The Old Malthouse, Clarence St, BATH, BA1 5NS. Tel: 0225-462602
	Tim Perry	Jubilee House, Chapel Rd, HOUNSLOW TW3 1XT. Tel: 01-577 3541
APL Impetus Ltd	Cedric Heddle	Rusper, Sandy Lane, Ivy Hatch, SEVENOAKS, Kent TN15 0PD Tel: 0732-885126
APL People	Jill Moss	The Old Malthouse, Clarence St, BATH, BA1 5NS. Tel: 0225-462602
APL Software	David Alis	Jubilee House, Chapel Rd, HOUNSLOW TW3 1XT Tel: 01-577 3541
APL Software Technology	John Hagger	14 Rosewood Avenue, Alveston, Bristol BS12 2PP Tel: 0454415737
	Per Hulfin	46 Vicarage Road, South Benfleet, Essex SS7 1PB Tel: 0374550501
Beta-plan APS	Kim Andreassen	Stengrade 75 , DK-3000 Helsingor, Denmark. Tel:45 2 21 48 48
Buckland Management Systems	John Buckland	Wastwood, 9 Grove Road, Camberley, Surrey, GU15 2DN Tel: 0278 684327
Butel Technology Ltd.	Mike Munro	Butel House, 3 Great West Rd., London W4 5QJ Tel: 01-995-1433
Anthony Camacho		2 Blenheim Road, St Albans, Herts AL1 4NR. Tel: St. Albans (0727) 60130

Paul Chapman		18, Trevelyan Road, London, SW17 9LN Tel: 01-767 4254
Cocking & Drury Ltd.	Romilly Cocking	180 Tottenham Court Road, LONDON, W1P 9LE Tel: 01 438 9481 Fax: 01-436 0524
Datatrade Ltd.	Tony Checkself	38 Billing Road, Northampton, NN1 5DQ. Tel: 0604-22289
Delphi Consultation	David Crossley	Church Green House, Stanford-in-the-Vale, Oxon SN7 8LQ. Tel: 03877-384
Dyadic Systems Ltd.	Peter Donnelly	Park House, The High Street, Alton, Hampshire. Tel: 0420-87024
E & S Associates	Frank Evans	19 Homesdale Road, Orpington, Kent BR5 1JS. Tel: 0689-24741
Farnell International Instruments Ltd.	R. Fairbairn	Jubilee House, Sandbeck Way, Wetherby, W. Yorks. Tel: 0937-61951
	Roger Attard	Davenport House, Bowers Way, Harpenden, Herts. Tel: 05827-69071
FASTCODE	Andrew Dickey	P.O. Box 281, Croton-on-Hudson, New York 10520, U.S.A. Tel:(914) 271-3200
General Software Ltd.	M.E. Martin	22 Russell Road, Northolt, Middx. UB5 4QS. Tel: 01-864 9537
H.M.W. Computing Ltd.	Stan Wilkinson	Hamilton House, 1 Temple Avenue, Victoria Embankment, LONDON EC4Y 0HA Tel: 01-353 4212 Telex: 926604 HAMHSEG Fax: 01-353 3325
HRH Systems	Dick Holt	Box 4496, Silver Spring, Maryland 20904
Ian A. Clark		9 Hill End, Frosterley, Bp. Auckland, Co. Durham DL13 2SX Tel: 038852-7190
IBM UK Ltd	Nat.Enq. Centre	414 Chiswick High Rd, London W4 5TF Tel: 01-747 0747
Inner Product Ltd.	Dominic Murphy	Eagle House, 73 Clapham Common Southside, London SW4 9DG. Tel: 01-873 3354
Interprocess Systems	Stella Chamberlain	9040 Roswell Road, Suite 690, Atlanta, Georgia 30350-1131 Tel: (404) 992-8400
Lloyd Savage Ltd	Philip Johnson	Cambridge House, Oxford Road, Uxbridge, Middx, UB8 2UD. Tel: 0895-59826
Mercia Software Ltd.	Gareth Brentnall	Aston Science Park, Love Lane, Birmingham B7 4BJ. Tel: 021-359 5096
MetaTechnics Systems	John Stenbridge	Unit 216, 62 Tritton Road, London, SE21 8DE. Tel: 01-670 7959
MicroAPL Ltd.	David Eastwood	South Bank Technopark, 90 London Road, LONDON SE1 6LN Tel: 01-922 8868
M.T.I.	Ray Cannon	7 Pine Wood, Sunbury-on-Thames, Middx. TW16 6SH Tel: 09327 80848
Parallax Systems Inc.	Kevin Weaver	60 West 9th Street, New York, New York 10011, U.S.A. Tel: 212-475-4001
Peter Cyriax Systems	Peter Cyriax	213 Goldhurst Terrace, London NW6 3ER Tel: 01-624 7013 (Answerphone) 0860-377963 (Mobile)
Portable Software	Richard Smith	60 Aberdeen Ave, Cambridge, Mass. U.S.A. 02138. Tel: 617-547-2918
QB On-Line Systems	Philip Bulmer	5 Sunney House, Portsmouth Rd Camberley, Surrey, GU15 1LB. Tel: 0276-20789
The Rochester Group	Robert Pullman	164 Pinnacle Rd., Rochester NY 14620 Tel:716-454-4541
Shandell Systems Ltd.	Maurice Shanahan	12 High Street, Chalfont St. Giles, Bucks HP8 4QA. Tel: 02407-2027
I.P.S.A. Ltd.	Mike Butler	7th Floor B Block, Coventry Point, Market Way, Coventry CV1 1EA Tel: 0203 256562
Sugar Mill Software Corp.	Lawrence H. Nitz	1180 Kika Place, Kailua, Hawaii 96734 Tel: (808) 261-7536
Tektronix UK Ltd.	Paul Morgan	Fourth Avenue, Globe Park, Marlow, Bucks SL7 1YD. Tel: 06284-6000
Univare	Eric Lescasse	15 Rue Erlanger, 75016 Paris, France Tel: (1) 45-27-20-61 Fax: (1) 45-27-20-71 Telex: 648348F UNIVARE
Warwick Univ.	Dr Andy Pole	Dept of Statistics, University of Warwick, Coventry, CV4 7AL Tel:0203-523066 x2158

APL
PRODUCT
REVIEWS

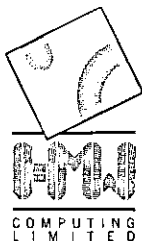


SOLUTIONS FOR THE CITY

We are a specialist firm of consultants with a prestigious list of clients including major City institutions.

For one of these clients we have recently developed a foreign exchange dealing system with the following features:- deal entry, deal enquiry, multi-currency position keeping, basket currency calculators, arbitrage calculators and ticket production.

If you are interested in foreign exchange dealing, treasury operations, portfolio management, commodity broking or similar areas we would be delighted to discuss how we could help. Call Stan Wilkinson, HMW COMPUTING LIMITED Hamilton House 1 Temple Avenue London EC4Y 0HA Telephone 01-353 4212



EXCITING POSITIONS AT THE LEADING EDGE

With top City institutions and leading Blue chip companies amongst our client list we have a lot to offer experienced APL programmers.

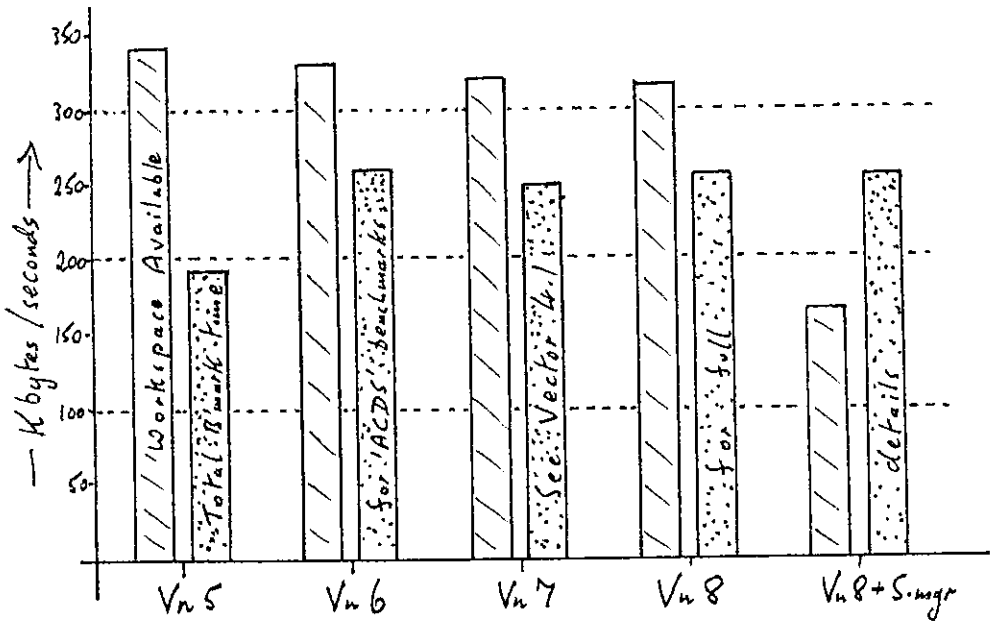
Call Stan Wilkinson to discuss your future.

HMW COMPUTING LIMITED
Hamilton House 1 Temple Avenue
London EC4Y 0HA Telephone 01-353 4212

Is APL*PLUS Getting Overweight?

by Adrian Smith

I started with APL*PLUS/PC back at release 4.0, running on a 256K IBM PC, with twin floppy drives. It took about 10 sec to load from disk, and it gave me just over 100K of free workspace. It also ran rather fast. The graph summarizes one view of progress since then:



The March of Progress?

In other words we have been forced into sacrificing speed and economy for a whole raft of new features that most of us could well do without. It isn't just APL that suffers from this disease; MS word (release 1.0) would run on similar hardware very happily, it now requires most of 640K and a hard disk. It is also a good deal slower than it was at the simple task like scrolling around your document.

The new session manager (avoidable with APL*PLUS/PC) is unfortunately an integral part of the II system. It is so gross that you have to shoehorn it to fit into a 500K machine: yes that's just for the session manager - the APL runs in high memory!! Let me make a few comments about its effectiveness (since all the STSC handouts concentrate only on the functionality):

- with the default keyboard layout, you need to hold down *three* keys to do that most obvious and common operation - erase to end of line. On the majority of PC keyboards, the three keys in question are so arranged that this has to be a two-handed operation!
- the APL function editor `EDIT` no longer indents labels and comments. If there is a way of turning this feature on, would STSC please tell me about it; I can't find anything obvious in the manuals.
- the session scrolling is a lot slower than it was in APL*PLUS/PC. I reckon that Ctrl+Up goes by faster on my plain old 8088 PC than on my whizzo 386 machine; what's more it stops when I take my finger off! The II system badly needs anti-skid braking!!
- the same problem occurs in the editor, only it is much nastier here. As an example, I set out to delete the last 20 lines from a 30-line variable; I held down Ctrl-Delete for just long enough to see the last unwanted line vanish, then was a little put out to see the editor swallow the remaining 10 lines, back up to the top! To add insult to injury, it then beeped at me several times to admonish me for my attempt at deleting the non-existent line[0].

This sort of thing is absolutely typical of many of today's software products. STSC are on record as saying that they listen to their users, and that their policy is to include everything they can directly into the interpreter. As with the Lotus Corporation, the result is a bit like 123-Version-3; full of features, but so slow that no-one will use it!

To summarize: if your user-environment has gotten so goddamn complicated that only someone with a Black Belt in Origami could drive it - don't try to alleviate the situation by adding another layer of confusion (in the shape of those useless pull-down menus); redesign it. Even better, buy a copy of Dyalog APL.

A Tale of Two File Systems: Myriade (for APL*PLUS/PC) and AFM (for APL2/PC)

reviewed by Adrian Smith

Introduction

Here are two file systems from well-known 3rd-party suppliers, aimed at improving on the in-built file handling of two of today's mainstream PC APL products. As far as possible, I will try to review them in parallel, although there are clearly some areas where the two packages cannot easily be compared.

The Suppliers

Myriade Data Systems is a French Canadian company, based in Quebec. They are a small, specialist supplier of APL software, with a good reputation for technical expertise. Their address is:

Myriade
3652 ch. St-Louis
Ste-Foy (Quebec)
CANADA G1W 1S9

AFM/PC is supplied by Interprocess Systems, of Atlanta, Georgia. They are a very well-established company in the mainframe APL marketplace, and again have a good reputation for technical excellence and software support. Their address is:

Interprocess Systems
9040 Roswell Road, Suite 690
Atlanta, Georgia 30350-1131

Objectives of the Packages

First Myriade. I suppose it has the harder task of the two, in that it sets out to improve upon an already established standard, in the APL*PLUS/PC component file system. If Myriade is to appeal to you, you must first have become dissatisfied in some way with the standard offering.

Some of the things Myriade does for you are:

- Indexed files. Of course we have all built little file systems which put the index on component-1, and use ^ . = in the workspace as our keying system. The emphasis should be on the word *little*; Myriade will handle keyed files of many Megabytes.
- high performance. Emulating indexes in APL is expensive in CPU time. Myriade aims to provide fast access to big files with a flexible keying system that fits well with the APL philosophy.
- strong emphasis on data integrity, with good backup and recovery procedures.
- generally better housekeeping. The APL*PLUS/PC file system is notorious for failing to recover space in files; space allocations can mushroom out of control alarmingly fast. Myriade takes great care to avoid file fragmentation.

If you are an APL*PLUS/PC user who has at some point encountered these restrictions and performance problems, then Myriade may well be for you.

AFM/PC is very simple to describe: it is a PC implementation of the well-known mainframe product which runs alongside VS APL and APL2. It provides APL2/PC users with:

- standard component files, and keyed component files.
- a simple mapping of AFM user numbers to DOS pathnames.
- good space management and recovery within files.

AFM/PC supports all the sensible commands from the mainframe system, except *FLOG* and *FSORT*; the documentation is simply a copy of the standard *AFM/AP User's Reference Manual*, with the addition of a 28-page PC User's Guide. If you are an APL*PLUS/PC user, who would like to test the IBM water, or an existing mainframe APL2 site wishing to migrate to the PC, then AFM/PC may be for you.

Installation and System Requirements

Myriade comes on a single 5-inch 360K disk. It may be copied to your hard disk, and then started for the first time to initialize the 'MFM Data Base'. You can specify the file size in multiples of 1Mbyte, and will then need to wait for around 1min per Mbyte while the database is initialized. A portion of MYRFM.EXE remains resident on completion.

When you start APL*PLUS/PC (version 7 or above) you must give a user number other than zero, otherwise you load APL as normal. Your $\square WA$ in a clear workspace will be reduced by 119K, although if you have access to an expanded memory manager, 64K of this can be loaded high. The database is automatically closed down on)OFF, but MYRFM remains resident until you reboot.

You will also need to load the workspace of supporting functions MYRFM.AWS; this requires a further 7K. It consists almost entirely of one-liners, and simply covers $\square CALL$ statements to the resident software. Since MYRFM manages the file space entirely on its own, you will need to use the supplied backup and recovery mechanisms, as all DOS sees is a single file which is likely to be much too big to fit on a single floppy disk!

AFM also comes on a single 360K disk, and can be copied to your hard disk as required. You must include AFM800 on the command line that starts APL2 (before AP120), and will probably need some of the functions from the supporting workspace. You should expect to lose 54K bytes of workspace.

Unlike Myriade, AFM creates normal DOS files (with a .AFM extension); obviously you should avoid naming other files in this way! Normal DOS backup and recovery applies.

Summary: both systems are painless to install, and performed correctly on the first time of asking. I suppose AFM was marginally the easier of the two, but both must score highly in this regard.

Features

Myriade is clearly more than just a fast component file system. Some of the extra features are:

- indexed files, using either character or integer keys. Character keys may be up to 121 bytes long. Components can be randomly created and updated.
- You can put up to 255 named APL variables per component. This allows easy modelling of relational systems, where each row of the table becomes a single file component, consisting of a set of variables (*NAME, AGE, SALARY* etc.).
- you can retrieve partial components (i.e. selected variables), optionally renaming the variables as they appear in the workspace.
- there is no effective limit on the number of components per file; files can grow up to 192Mbyte, spanning multiple DOS volumes as required. Space is recovered automatically, $\square FDUP$ is a thing of the past!

AFM is a straightforward component file system, with the addition of keyed access files (the maximum key length is 30 characters). It is effectively a single-user version of the mainframe system, but without some of the peripheral and administrative aids that come with the mainframe software. It can be run over a LAN environment (in which case the use of access codes, file holds etc. would be required); unfortunately I have not been able to test out any of this, as I do not have a LAN to play with!

Performance

Basically, Myriade scores heavily over APL*PLUS/PC as files get bigger. For a decent size file (say 5000 components) it will add a new component in under 1 sec, rather than 8 sec when a 20-character key is used to address each component. Retrieval is in under 0.5 sec (vs 1 sec) and deletion in 0.5 sec (vs 2.5 sec). When updating an existing component with a larger data value, there is again a factor of around 3, but of course the APL*PLUS/PC file requires regular use of `FDUP`, otherwise it quickly fills up the hard disk! The essential difference is that MFAM has a very flat performance curve, whereas the built-in file system is very dependent on file size.

All I can say about AFM is that it feels very comparable to the built-in file handling, via the library manager AP211. I don't think it is sensible at this stage to try and set up any serious benchmarks, as I don't really have anything to compare AFM against. If I get the time, I will code up some of my APL*PLUS/PC cover functions in AFM, and check out APL*PLUS/PC vs APL*PLUS II vs APL2/PC 32-bit. Watch this space.

Documentation

Myriade scores about 5 out of 10 here. Everything is there, but it is clearly a translation from a French original, and the style is odd in places. Also the quality of reproduction (of typescript rather than typeset material) is patchy. However it is complete, and with a little persistence, you can find what you need to know.

The AFM documentation is absolutely first class. I must admit to a slight bias here, because I have used and liked their mainframe system for several years, and have modelled much of my internal utility documentation on their style. In fact we used a page from the AFM Reference Manual as the model for the ASL manual at the Greg-y-Nog conference! You get a straight reprint of their mainframe reference manual, and a well typeset (A4) addendum describing specific features of the PC implementation.

Summary

These are both good products, and deserve to succeed in their respective markets. Myriade is to some extent a niche product, as it only shows its best form when dealing with really large files, and complex keyed access. I really don't know whether it is sensible to think of APL when designing such systems; to some extent the availability of a high-performance file manager swings the equation more in APL's favour. I suppose you also ought to worry about committing yourself to a 3rd party add-in, with no guarantee of support for future versions of APL*PLUS/PC, or of APL*PLUS II.

AFM provides a basic service for APL2/PC, just as it does for VS APL on the mainframe. If you are serious about developing APL systems on this PC product (remembering that you can package the resulting system at no extra charge) you should buy this product. Period (as they say in the States!).

Availability

AFM/PC works with DOS 3.1 or later, and with APL2/PC 1.0 (or later) in either 16-bit or 32-bit modes. It is priced at \$175 with volume discounts available. For more details, contact Stella Chamberlain on (404) 992-8400, or write to Interprocess Systems.

Myriade requires DOS 3.0 or later (performance is claimed to be much better under DOS 4.0) and APL*PLUS/PC 7.0 or higher. MFM Version 1.0 for APL*PLUS/PC sells for 375 CAN\$ from: Myriade, (418) 658-1275, Canada.

Ken Iverson with Arthur Whitney



photographed by Rob Hodgkinson at APL89

RECENT MEETINGS

This section of VECTOR is intended to document the seminars given at recent meetings of the association; it is of particular value to members who live away from London. It also covers other selected events which may be of general interest to the APL community.

If you would like to speak at one of the regular British APL Association seminars, please ring the Activities Officer (address on inside back cover) who will respond enthusiastically to your offer.

Notes on Recent Meetings

Introduction by Adrian Smith

Greg-y-Nog

Thanks to Alan Sykes for his notes on the APL Statistics Library meeting at Greg-y-Nog. Greg-y-Nog Hall was one of those places with a very special atmosphere; the kind of location which somehow encourages people to agree with each other and just get on with the job in hand! I felt that we achieved a remarkable amount in a very short space of time, and had some good fun in between (lyrics and music enclosed as promised).

The photos are by myself; I hope no-one feels grossly mis-represented! If anyone has any improvements on the captions (such as 'Spot the Balls' for the croquet scene) please forward them to the editor, who promises to print anything which is not actually libellous.

Alan's roundup is followed by Tony O'Hagan's more formal summary of the conference. This was presented to the British APL Association committee, and has been agreed as the way to proceed with the ASL project. We now have a programmer in part-time employment, and she is already well into the coding stage of the graphics section of the bottom shelf.

The Field Trip to British Airways

Put not your trust in Barcos!! The projector made a splendid *BANG* as the engineer (having just completed the setting up) dropped the metal stay for the cover into the power supply! Apart from this unfortunate accident (which was recovered from with remarkable speed and calm), a most interesting day. Thanks to all at BA for setting it up; I personally found it well worth the extra distance.

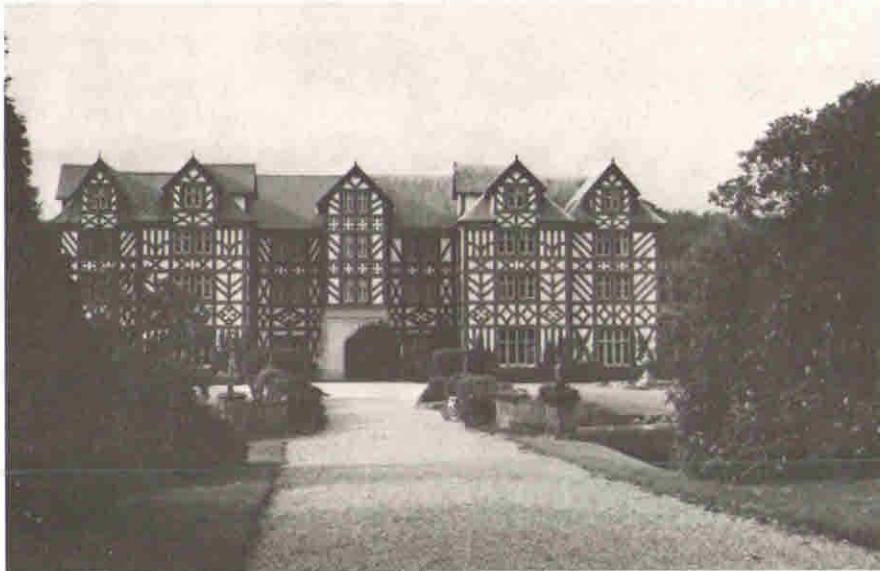
APL in Finance

Both speakers have provided full transcripts of their talks; need I say more!! I was very sorry to miss this meeting, so it is particularly pleasing to have the written record in full. Maybe I was unduly pessimistic in my Editorial; from Peter's notes I get the feeling that there may yet be a niche for APL in the corporate mainstream.

The ASL Conference

A report by Alan Sykes

The APL Statistics Library Project (ASL for short) was launched with a conference, held at the University of Wales Conference Centre at Gregynog (pronounced Gre-gun-ug ... Ed) near Newtown from the 18th to the 20th of September. It was attended by 29 people, 18 from academia and 11 from industry and commerce. All had a splendid time working hard during the day, and relaxing/imbibing in the evening. All agreed that it was a very successful event. So what did we set out to do, what did we achieve and what has happened as a consequence?

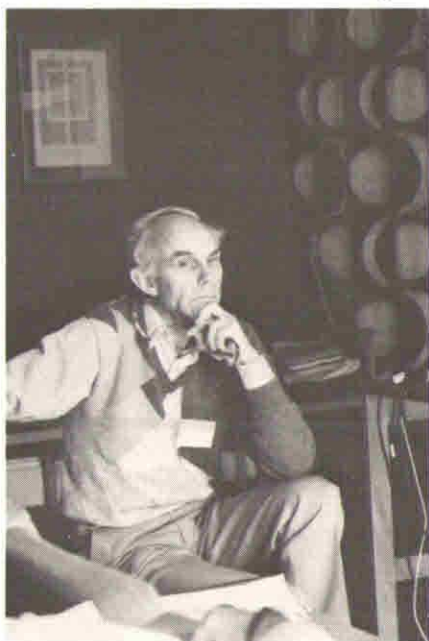


Greg-y-Nog Hall From the Gardens

The aim of the conference was to discuss the feasibility of the ASL project, to listen to talks and discuss proposals on the content and methodology of the project, and to set up a management structure and appropriate working parties for ASL - quite a task for just over 48 hours. However all participants worked very hard and considerable progress was made towards these aims.

Tony O'Hagan (of the Department of Statistics at Warwick University) started the conference off with a brief introduction to ASL. Tony suggested the library should consist of 'shelves' - and we were quickly led by Norman Thomson into a comprehensive suggestion for the 'bottom shelf' together with a rather daunting list of urgent topics for consideration under the general headings of:

1. Management
2. Programming/Computing
3. Statistics/Numerical Analysis
4. Further Shelves in the Library



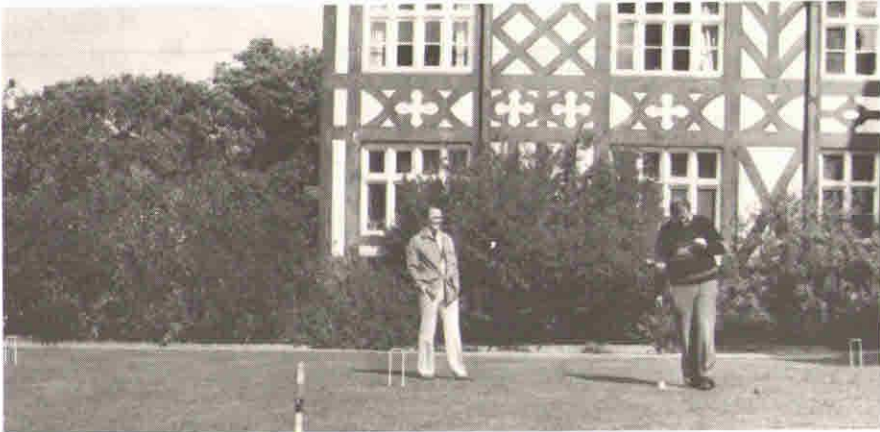
... while Norman Thomson Listens intently!



Tony O'Hagan Leads off the Conference

Not surprisingly, tea was required after we had plunged so willingly in at the deep end. (Gregynog teas are a worthy reward for any conference session!) Duly refreshed, we returned to the conference room to be further entertained and enlightened, this time by Adrian Smith on a topic he is well qualified to propound - what constitutes good APL code. It was a salutary experience for us academic APLers as we realized how many of the bad coding methods that Adrian exposed were endemic in our hitherto prized APL functions. After dinner, Alan Mayer presented some thoughts

on the teaching of Probability and Statistics using APL, rounding off day one of the conference.



Relaxing (?!!) on the Croquet Lawn after Tea

The next day was divided basically into two different types of sessions. In the morning Tony O'Hagan presented his thoughts on how the core graphics functions in the bottom shelf should be constructed, based on his experience with 'WAGS' at Warwick. Jake Ansell and I then tackled the favourite APL statistics topic - that of Regression Analysis.

By contrast, the afternoon was devoted entirely to parallel workshops on the topics so far presented, namely the Bottom Shelf, APL Coding Standards, Graphics and Regression. As a bit of light relief, a musical entertainment (for which I must take full responsibility!) finished the day off.



Martyn Adams of MetaPraxis

The final day of the conference consisted of suggestions for further ASL shelves - on Time Series (given by Andy Pole) and on Reliability Modelling (by Jake Ansell). To balance the statistical input, two presentations were given on constructing packages in APL, by Martyn Adams and Maurice Jordan. And finally Tony O'Hagan brought all the conference contributions together in a planning session.

So what conclusions did we come to, and what decisions did we make? Everybody agreed that ASL was a worthwhile project and should go ahead, though we were all well aware of the enormity of the task, particularly as so much of the input would require that rare commodity these days - participants' spare time! The detailed decisions were later written up by Tony and presented to the BAA for their approval in the following article entitled 'ASL after Gregynog'.

Thanks must go to the BAA for their unanimous support of the ASL conference and project, and in particular to David Eastwood who was always available for advice when it was most needed. But above all I must give thanks to all who so willingly participated in the conference - if their enthusiasm is maintained and appropriately channelled into productive work, then the success of ASL is a realistic target.



A Bard's Eye View of the Proceedings

ASL after Gregynog

by Tony O'Hagan

The ASL conference at Gregynog proved a tremendous success. In two and a half days, the whole ASL project has been given a new and much clearer shape and a greatly increased impetus.

Decisions Made

1. ASL should go ahead in the form and according to the timescale described below.
2. ASL should consist of APL functions primarily to perform statistical calculations, with basic facilities for displaying the results (including some graphics), but without attempting to develop a sophisticated user interface.
3. ASL should be independent of commercial interests, and the ASL functions should be made freely available for only a nominal charge.
4. ASL should be documented in book form, giving detailed information on the functions including full code. This book will thereby set the standard for statistical computations in APL. The functions should also be made available in machine-readable forms.
5. The form of ASL described above is intended to be the most useful resource for statisticians who are already familiar with APL. Packaging ASL with a user-friendly interface would not be helpful for such an audience, and would even be an impediment to realising the power and flexibility of APL. However, it is vital to reach wider audiences. For the statistician who is not familiar with APL, a tutorial text should be written to explain how to use ASL for common statistical analyses. If this book is well done, the power and elegance should tempt the reader to learn APL.
6. The even wider audience of users of statistics packages will need a user-friendly front end and more eye-catching graphics. There should be a real commercial interest for companies to provide such a package. The market is large, and the basic computational code will have been done for them already. Commercial exploitations of ASL for this and other markets should be licensed so as to provide income to continue the development of ASL. The financial management should be handled through the BAA.

7. The language for ASL should consist of ISO standard APL ('first generation') plus nested arrays. The use of nested arrays permitted in ASL should be carefully defined so as to be compatible with as many 'second generation' interpreters as possible. This means that ASL will not be compatible with the Sharp approach, but there should be no other problems, and it was felt that to confine ASL to ISO APL would be short-sighted. However, nested arrays are inevitably more complicated for the user, therefore they should only be used where they confer a distinct advantage over the simple arrays.
8. ASL should not exclude any branches or methods of reputable statistical analysis. As a consequence, ASL is potentially very big indeed. Many of the decisions which follow reflect this fact.
9. ASL will need a variety of high-level library management and information tools. Users will need to obtain information about what functions are available for specific tasks, and to be able to extract those functions together with all other functions called by them. Such 'anthologies' of functions, which form a coherent group to perform a particular kind of analysis, will cut across any structuring of the library into 'shelves' or 'volumes'. Standard 'anthologies' should be readily available for common needs.
10. A number of guidelines for good coding practices were agreed. The need for naming conventions in a library was stressed. A full set of standards should be defined soon.
11. A large, highly interconnected library will be very vulnerable to bugs introduced when functions are changed. Change is inevitable as ASL grows, but changes to existing functions must be rigidly controlled. It is particularly important that the utility functions in the bottom shelf be got right from the start.
12. The time-scale for ASL needs to be realistic. The programme of work that was initially submitted to the BAA was reviewed. It was decided that more time should be allowed in the first months for establishing the fundamental structure, standards, development and management tools, etc., in the light of the increased size that is now envisaged for ASL. The bottom shelf and regression shelf should be written as previously envisaged, although it is now recognised that in those forms they will be far from complete (Completeness will never be a property of ASL!) The extra work on fundamentals will delay the regression shelf by three months, so that the programme is now as follows.

October-December 1989 - Finalise standards. Write development and management tools. Write preliminary version of the bottom shelf.

January-March 1990 - Test and prove bottom shelf, leading to selected release. Specify regression shelf.

April-June 1990 - Finalise bottom shelf for general release. Write preliminary version of regression shelf.

July-September 1990 - Test and prove regression shelf, leading to selected release.

13. The bottom shelf should: (a) provide a basic statistical analysis capability at a level appropriate to a first course in statistics, e.g. 'A' level or first year university, and (b) include basic utility functions of all kinds, for use in other shelves. As for ASL, as a whole, the emphasis is on numerical functions, supported by basic display facilities.
14. The bottom shelf will include graphics functions written in a graphics primitive language based on the Warwick APL Graphics Standard (WAGS). Implementations of WAGS should be developed to drive a range of graphics devices from a choice of interpreters. However, implementation should be left primarily to vendors, since it will be in their commercial interests to make ASL usable with their products.
15. ASL will be coordinated by a small group of people. It was not clear that 'committee' would be an appropriate word for this group in the absence of any formal constitution, so I will call them the Management Team. The following were nominated.

Tony O'Hagan - Chairman
 Jake Ansell - Manager
 Alan Sykes - Vice Chairman
 David Eastwood, John Searle - BAA Representatives.

16. The specifications and development of each part of ASL will be the responsibility of a small (2 or 3 person) Working Party, reporting to the Management Team. The Manager will be responsible for the Programmer, providing programming support for the Working Parties. The following Working Parties were envisaged as being needed at the outset:

Standards and Tools
 Bottom Shelf Statistics
 Bottom Shelf Graphics
 WAGS Implementation - —

17. The involvement of the wider statistics/APL community in ASL is vital. Communication will be through the APL Statistics User Group. Major developments should be reported in VECTOR.

APL is a *Very* Funny Language

as sung by The Alan Sykes Trio

1. A P L ers use these funny symbols - upgrade,
downgrade, rho and many more.
When you get them talking and drinking
God, they really are an awful bore!
(with a jot-dot here and a jot-dot there - here a jot, there a jot,
everywhere a jot-dot...)

Chorus

*APL is a very funny language - it is one of the very best;
Won't you come and join the B A A?*

2. Adrian Smith says you must stop all looping:
always go to *Ex i t* when you end
Avoid all brackets, design before you write it -
That's enough to drive you round the bend.

Chorus

*APL is a very funny language - it is one of the very best;
Won't you come and join the B A A?*

3. Norman's tried to inculcate some standards
Do you mean by *MEAN* what I mean too?
On his shelf are lots of lovely functions
How to use them, no we've not a clue

Chorus

*APL is a very funny language - it is one of the very best;
Won't you come and join the B A A?*

4. Alan Sykes is definitely the key-man
He has brought us here to Greg-y-Nog
Perhaps we'd better take him for a walk now
Get him stuck right in a good Welsh bog

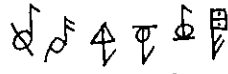
Chorus

*APL is a very funny language - it is one of the very best;
Won't you come and join the B A A?*

Won't you come and join the B A A

Kate Morris/Alex Sykas

The APL Song

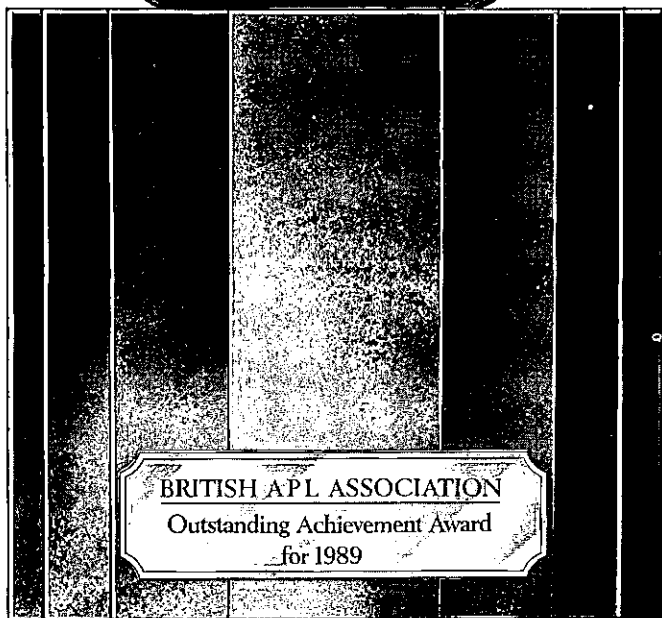
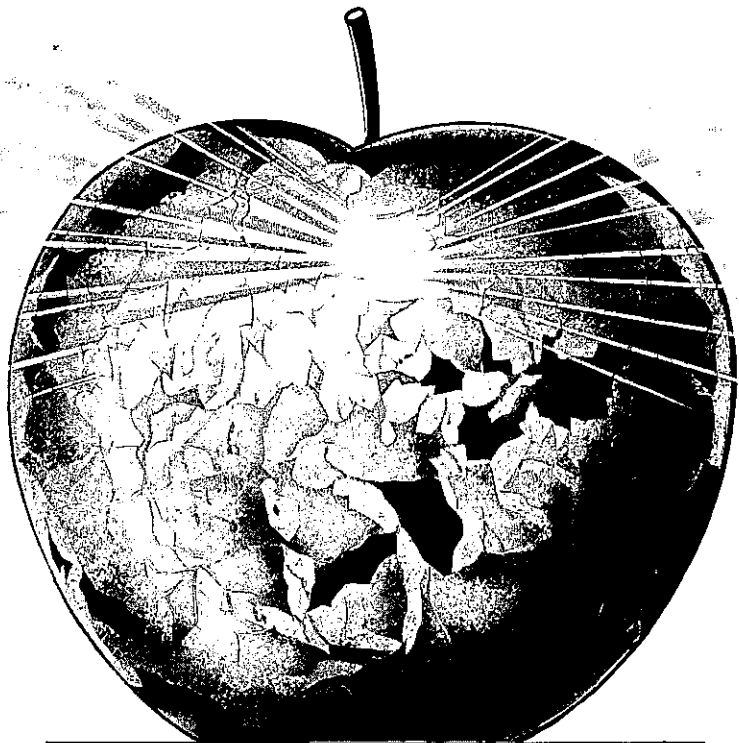


Chorus

A P L is a Very funny language it is one of the Very best Wou't you come foad

join the B-A-A Fine

Solo Chorus



USE YOUR APL SKILLS WITH AN AWARD WINNING TEAM

British Airways has just won the APL Association Achievement Award for 1989 — a recognition of our contribution to the development across APL for use in worldwide telecommunications networks.

So what better time than now for you to be part of our innovative IT team at Heathrow?

By using advanced technology to collect, process and analyse complex marketing and financial data, you will make a real contribution to the commercial success of British Airways. This involves developing your APL skills on projects ranging from decision support systems for our international sales force to Executive Information Systems for our Company Directors.

ANALYST PROGRAMMER

£14-20K package

With at least 4 years' commercial computing experience, you will ideally have a mathematical or Operational Research background and be familiar with decision support systems. A knowledge of VM/CMS, C, PL/1, SQL or modular programming would be an advantage. Excellent problem-solving skills are essential. Ref: JDW/1908/210.

PROGRAMMER

£10-17K package

With at least 1 year's commercial computing experience, you will ideally have a mathematical or Operational Research background and a knowledge of VM/CMS, C, or PL/1. Ref: JDW/1908/211.

For both posts you will need to be self-motivated and able to work well on your own or as part of a team. There are outstanding opportunities to advance into team leadership, business analysis, or a specialised technical area.

The rewards, like the challenges, are substantial. Apart from a performance related salary, you will receive a profit share, holiday bonus, holiday travel opportunities, outstanding sports and social facilities — and the kind of career development opportunities that only a world leading IT operation can provide.

Interested? Then please write with a full c.v., quoting the appropriate reference number, to: Selection & Assessment, British Airways Plc, "Meadowbank", PO Box 59, Hounslow TW5 9QX.

The logo consists of the words "BRITISH AIRWAYS" in a bold, white, sans-serif font, stacked vertically. Below the text is a white diagonal swoosh that starts from the bottom left and points towards the top right, set against a black rectangular background.

Field Trip to British Airways: 27 October 1989

notes by Sylvia Camacho

Introduction

British Airways staff gave presentations and demonstrations of four of their mainframe computer systems to about 50 members of the Association.

AIMS: Airline Information Management System

AIMS has approximately 120 users. It was originally intended for top management only but has extended down the chain. It was a Marketing Department initiative but has since widened its scope. The system has about 30 menus covering revenue, traffic, manpower, timetables, market share and a snapshot summary of BA performance updated daily, weekly and monthly. PF keys are extensively used by expert users for a quick change of menu.

Statistics are presented as absolute values or PF4 converts to graphics. Updating is fast - last weeks performance is ready by Monday morning. All screens are in full colour and the colours are given a common meaning (e.g. green for actual, blue for budget and so on) over all BA systems.

The information is shown at all levels of breakdown of the hierarchy and broken down for remote sites. The Manpower Menu gives access to costs, job breakdown, overtime etc. The Customer service menu includes statistics on punctuality. Marketing includes capacity, future orders, Budgets, Audit reports and 5 Year Plan and so on.

The top 150 managers are given a presentation of half year result figures but none of this information is printed and normal access is limited to a few users. All menus are independently controlled to give various users access to some and not other menus.

The data is collected from many sources including public airline databases (for competitor information) and aviation news. Users can extract information to highlight trends e.g. Japan is a big growth market. The system includes a mailbox for sending messages to other users and for public news items and an on-line phone book.

The system users are charged quarterly based on a proportion of the data storage (130 Megabytes) and support staff (2). Training in the use of the system takes one hour. The data files are made available to other computer users where appropriate.

LOOK (acronym unknown)

This is a database of 20 million records, used by 40 Departments.

The system is an historical product database. The product is passenger-kilometres broken down by class, aircraft type, route/legs, revenue, costs, load factors and so on.

LOOK is an enquiry, extract, match and reporting system based on a data dictionary of the fields available. PF1 from any point gives help with the syntax to be used to manipulate the data. The system is entirely full-screen. New fields can be created from user formulae and used in any enquiry. Any suitable output can be shown in graphical form. Screens are full colour. All sessions are logged and enquiries can be saved in order to run them automatically against new data. Extracts can be exported as new files.

MARS: Multi Dimensional Array Spreadsheet

This is a mainframe based multi-dimensional spreadsheet. The demonstration was of a Revenue/Budget array of 5 dimensions - passenger revenue by station-pairs (500 flight legs) by 129 sales areas by 5 classes by 12 months. This 7.74 million numbers would require 31 Megabytes to store but it is a sparse array and reduces to 4 Megabytes when techniques are used to remove zeros.

MARS is combined with a flexible editor MEDAL which allows formulae to be input to add calculated fields. During what-if enquiries field changes are distributed immediately to show the effect of the change. The editor allows multi field changes from simple input (e.g. increase dimension by a factor or change sub-totals and distribute back pro-rata). Values can be shown as % or scaled to change field width and can be left justified for input or right justified for display. Changes specified for more than one field can be applied while holding selected fields constant. A graphics package is incorporated. Consolidations are selective and the selection can be held for future use. There are some pre-defined system consolidations.

MARS is a general purpose package/file structure. BA have used up to 11 dimensions. Extracts from the dimensions can be laminated to give new file

structures. There is a sort facility. There is a development in hand to capture a command set from the Session Manager and thus allow enquiries to be run in batch mode.

FILLUP (acronym too contrived to be worth remembering)

BA spend £500 million a year on aviation fuel and have contracts with many suppliers to deliver the fuel directly to the aircraft. The contract is legally binding to take a given quantity of fuel to within 0.05% of contracted quantity, regardless of price.

The manual schedule for the deliveries to the aircraft for a given timetable used to take 4 days and now takes 2 hours. This will make it possible to reduce the span of the contracts from half-yearly to some shorter period and already makes it possible to react much faster to price changes to optimize the take up from cheaper suppliers while staying within contract constraints. The system produces a daily or on demand schedule for each supplier giving time, position and aircraft type and capacity. It matches smaller suppliers to small aircraft and spreads the suppliers load as evenly as possible over the day to optimise their manpower and tanker usage. This makes for happier suppliers and smoother running of flights.

Conclusion

The impression we received was that BA had a good commitment to modelling systems which they have been developing over at least 7 years. They emphasised a concept they called 'branding'. This is a means of helping the users to identify a particular computer service with a name e.g. AIMS, LOOK, MARS, FILLUP. The users associate this with a particular style of facility and support and know where to go for information or to ask for additions. BA also try to standardize certain aspects of the systems such as the significance of colours and certain screen styles and key usages. It was notable that these systems had been refined over the years instead of being replaced. This probably arises from a good basic design together with a supportive computer department.

B.O.S. - A Management System for Multicurrency Accounting

by Peter Branson (Electronic Data Systems)

About the Company

Electronic Data Systems (EDS) was originally an independent U.S. company, but is now a wholly owned subsidiary of General Motors; it provides many services, including software and hardware, integrated systems, communications facilities, and manufacture of various electronic systems. The company has many mainframe sites around the world, linked into a major network, and it also operates client mainframe sites on contract.

The larger part of EDS (about 45,000 employees) deals exclusively with North America, but all other international business is handled by EDS International (about 5,000 employees) which has its HQ in London. Although EDS's largest client is GM itself, it has a large and growing list of external clients, including industry, commerce, banks, airlines, governments etc.

B.O.S. - Budget and Outlook System

For accounting purposes, EDSI uses the well-known MSA (Management Sciences of America) multicurrency accounting suite. This is run under CICS on an IBM mainframe, and is used both for internal accounts and as a service for external clients. Taking the former, all EDSI account transactions worldwide go to the MSA accounts receivable or payable, and then to the General Ledger. Cost centre totals for the day are then sent to BOS by overnight batch file, and are held in BOS for the current month and 36 months historically.

The 'budget' is a normal January - December monthly budget which remains fixed for the year from January 1st. Because a fixed annual budget can soon get out of date for volatile accounts, BOS also provides an 'outlook' system; this is a forward rolling budget which always covers 12 months ahead, and which account managers can update each month.

BOS (like MSA) incorporates a hierarchical coding setup which defines how cost centre totals are to be aggregated up the various levels of the company structure (department, region, country, division etc.). The system also allows for expenses accruing in overhead cost centres to be allocated according to rules (which can be adapted) to the direct expenses of contract cost centres. Essentially, BOS calculates revenue, expenses, gross contribution and pre-tax profit, plus many intermediate quantities, at both cost centre level and any or all of the levels above - right up to the company total.

At the lower levels, results are available in either local currencies or US Dollars, but when aggregating above country level, all results are in a common currency, e.g. dollars. With time-series data, changes in exchange rate can distort comparisons, so the system also holds 'restated' amounts (i.e. converted to a common exchange rate) to give more appropriate time-series comparisons.

BOS is a menu-driven full-screen APL system, with 100+ screens nested up to 5 deep. In EDSI, all account managers have to use it, and there are currently some 400 users worldwide. A variety of results are available: the month-end 'close' report gives a detailed summary of the month's business; there is also a large selection of comparisons with budget or outlooks; rolling totals, year to date etc. plus various projections. Data is available either by display (in many cases graphically) or print, or can be downloaded to PC spreadsheets.

Since the underlying MSA data which feeds BOS also satisfies the statutory accounting requirements of the host countries, the company is assured that its financial reporting is completely self-consistent across all levels of management.

In passing, I would like to make a brief mention of the EDS dress and conduct code, since many hold the outmoded idea that this is very rigid and militaristic.

Although there may have been some truth in this historically, it is certainly not the case now; working conditions are exactly what you would expect in any major company dealing with outside clients, and people are very friendly at all levels.

An excellent point in EDS's favour is that it has no age restrictions on employment, which has meant that I have been able to obtain a full-time job using APL at only a few years short of normal retirement age!

APL Aspects of B.O.S.

Written originally in IBM's VS APL, it is now running under APL2, and some of the language enhancements are being found very useful. For example the hierarchical node linkage is conveniently held as a nested vector of numeric vectors. Other nested arrays are being used to simplify slabs of arithmetic, as well as to group together variables so as to minimise the (undesirable)

use of semi-globals. The new *TIME* function under release 1.3 is excellent for code optimisation.

Each end-user's data is different, and this is presently stored as sequential (QSAM) files; there can be thousands of these, giving a heavy I/O load. Work is in hand to replace these with a very large VSAM file, using an EDS-written assembler routine for file compression. This is expected to show substantial savings. (*Have you looked at AFM? Ed.*)

We also intend to use the new 'packaged' workspace facility, and have installed Release 1.3 to do this. However, initial tests showed that for overnight batch consolidation runs the CPU time was sometimes increased by up to 30%. This was quite unexpected, and IBM are investigating as a matter of urgency. The problem may be due to heavy file I/O, but other users of Release 1.3 would be well advised to do some careful timing on intensive applications.

BOS is by no means static, and further work is planned. Internally, we plan to use DB2 to simplify some of the fairly complex relational aspects. We are also likely to move towards downloading BOS calculations to the PC, using the mainframe only as a file-server and for batch work. In terms of function, the end users keep asking for new features to be added.

Finally, why APL? (APL is not in widespread use within EDS.) A few years ago, most of this work was done in ADRS spreadsheets, and then on PC spreadsheets. However the inadequacy of PC spreadsheets for heavy calculation soon became clear. Fortunately, an embryonic budget system in APL (originating from GM in Germany) already existed, and it was decided to build on this, starting in 1986. The users were given a functional system quite quickly, and this has been extended and improved quite rapidly. The managers who 'own' the system have all sorts of housekeeping utility screens to maintain user-tables, hierarchy tables etc. They are also quite happy to have simple changes put in place overnight, and not to have to wait too long for bigger things.

Well, that's APL, isn't it!

APL for Financial Calculations in Insurance November 24th at Over-Seas House

by D.O.Forfar

In 1982, in a contribution to a colloquium on non-life insurance (XVI ASTIN Colloquium, Liege), the Swiss actuary J.A. Bardola concluded that APL was a very attractive programming language for non-life insurance problems.

In a paper to the International Congress of Actuaries (Sydney, 1984), the French Actuary L. Moreau described the use of APL in the French insurance company Assurances Generales de France (AGF) - one of the largest insurance companies in France. The uses to which APL was put were:

- 1 the development of APL functions to calculate all actuarial functions - the 'pocket actuary';
- 2 a system of forecasting financial accounts whereby the accounts for subsequent years were simulated on the basis of certain assumptions;
- 3 a model simulating the 'life' of 10,000 insurance policies and calculating the profitability of the portfolio of policies;
- 4 the checking of mortality tables;
- 5 the analysis of the mortality and surrender experience of policies.

A book entitled "APL, the Language and its Actuarial Applications", by S. Stiers, M.J. Goovaerts and J.de Kerf (North Holland) was published in 1987. One of the authors, M.J. Goovaerts, is Professor of Actuarial Mathematics at the University of Leuven (Belgium) and at the University of Amsterdam. The book contains a wealth of examples of the application of APL to problems in finance, life and non-life insurance, credibility theory, statistics, numerical analysis etc.

Papers on the application of APL to insurance problems have also appeared in the journal of the German Actuarial Society. A paper will soon be appearing in the Journal of the Institute of Actuaries (by the author of this note).

There are now a substantial number of financial institutions (insurance companies, insurance brokers, banks, hire purchase companies etc.) in the UK using APL. As will be seen from the references referred to above there is considerable use in Europe of APL for insurance calculations; indeed its use on the continent may well exceed that in the UK.

The syntax of APL lends itself very well to financial problems - in a sense APL may be described as a natural language for the solution of financial problems. The vector, matrix, multi-dimensional array structure of the language, the special symbols and the use of simple functions as building blocks to more complex functions can be exploited in powerful ways.

We may take a very simple example as an illustration.

We suppose that an insurance company undertakes to make annual payments of £1 on an annuity policy at the end of each of the next n years.

Years	1	2	3	n
Payments	£1	£1	£1		£1

For those readers who remember their compound interest the present value of these payments is:

$$\sum_{r=1}^n \frac{1}{(1+i)^r}$$

where i is the rate of interest, or in APL:

$$+ / (\div 1 + I) * 1 N$$

If the payments, instead of being £1 at the end of each of the years, are (a1,a2,a3.....an) then the value of the payments is $+ / A * (\div 1 + I) * 1 N$

If the rate of interest is not i, but varies from year to year, being i1,i2,.....in during each of the next n years, then the same APL expression holds with I being the vector (i1,i2,.....in).

The above is a very simple example of the great economy of expression, without loss of clarity that can be obtained for the solution of financial problems in APL.

My own insurance company (the Scottish Widows' Fund) uses APL in the Actuarial Department and in the Pensions Department. The Actuarial Department makes extensive use of APL as follows:

- 1 Calculation of actuarial functions (the 'pocket actuary').

- 2 Calculation of premium rates on different actuarial bases - the actuarial assumptions underlying the rates may be changed interactively and the effect on the rates can be seen immediately.
- 3 Profit testing of unit-linked products.
- 4 Solution of miscellaneous actuarial problems, by developing appropriate APL functions. These can usually be developed very quickly.
- 5 A system is available to our branch offices to enable them to request from their branch terminals premium and annuity rates. The system is menu driven and written (in APL) and maintained by the Actuarial Department - with no reference to IT Department apart from the general security/supervisory role. The rates are calculated, when requested, directly from the underlying actuarial bases. Printed books of tables of rates would be so voluminous as to be impractical.
- 6 Modelling of the office as a whole. The assets and liabilities of the office are entered into the model which can be run forward either deterministically or stochastically on different future financial scenarios and different bonus strategies investigated and their effect on statutory solvency ratios determined etc.

It has been our experience that actuarial students have had little difficulty learning APL. They have enjoyed using it as they have recognised its power to solve financial problems. The use of APL in the Actuarial Department has enabled productivity in certain areas of the department to increase by a factor of between two and three.

GENERAL ARTICLES

This section of VECTOR is oriented towards readers who may neither know APL, nor be interested in learning it. However we hope you are curious about how, under the right conditions, such impressive results can emerge so quickly from APL programmers

Variance of an Arithmetic Expression - An Example of Symbolic Computation and Recursion

by Tony O'Hagan, University of Warwick

Background

I recently had occasion to indulge myself in some interesting APL, exploiting APL's power in recursion and in manipulating character information. For the purposes of this article, I have greatly simplified the problem, and hence the solution, but even in this reduced form I think it raises some interesting issues.

The Problem

I was interested in the cost, C , of a certain, rather large project. C was not known precisely and was therefore regarded as a random quantity. In order to estimate it, I required its expectation $E(C)$ and variance $V(C)$. The following (fictitious but not entirely unrealistic) example may help to explain how such a problem may arise.

Suppose that a new section is planned for an existing motorway, connecting it to another. C is the total construction cost. In thinking about C , we will naturally decompose it into identifiable sources of cost. There will be a new junction at the connection with the other motorway, at a cost J . We will also need to replace the existing terminal of this motorway with a proper junction, but the existing work makes the cost of this junction less than J , say $J \times P$, where P is the proportion of new work required. There will also be a bridge over the new section at a cost B . Then the cost of actually laying the road is $K \times L$, where K is the number of kilometres involved and L is the laying cost per kilometre. (We structure this term in this way because we have good information from previous motorway projects about L .)

Finally, we could suppose that a rebate of proportion Q of the last cost is available (perhaps from the European community), so we subtract $K \times L \times Q$. We have constructed a *model* for C :

$$C = J + (J \times P) + B + (K \times L) - (K \times L \times Q)$$

Typically, none of the quantities J , P , B , K , L and Q will be known exactly in advance, but we will have information in the form of means and variances like $E(J)$ and $V(J)$. Some quantities, like Q or K , may be more or less predictable, and so have small variances, but others will be very difficult to predict and will have relatively much larger variances. We can also assume them to be essentially independent of each other. (We can define C as a cost in 'constant pounds' to eliminate inflation, which would otherwise cause quantities like J and B to be correlated.)

In general, then, we have a model relating C to various other random variables, which are statistically independent and whose expectations and variances are known. Then finding $E(C)$ and $V(C)$ reduces to applying some standard formulae for sums, differences and products of random variables. First, for any X and Y :

$$\begin{aligned} E(X+Y) &= E(X) + E(Y) & V(X+Y) &= V(X) + V(Y) + 2C(X, Y) \\ E(X-Y) &= E(X) - E(Y) & V(X-Y) &= V(X) + V(Y) - 2C(X, Y) \end{aligned}$$

where $C(X, Y)$ is the covariance between X and Y , and is zero if X and Y are independent. Also, if X and Y are independent:

$$\begin{aligned} E(X \times Y) &= E(X) \times E(Y) \\ V(X \times Y) &= (V(X) \times E(Y)^2) + (E(X)^2 \times V(Y)) + (V(X) \times V(Y)) \end{aligned}$$

(The last is not very well known but quite easy to prove.) In applying these formulae to our example model, we find only one complication. Although individual variables in the model are independent, we want to apply the formula for $V(X+Y)$ when X and Y may be products. Now $X=J$ and $Y=J \times P$ are not independent because they have J in common. We need another general result ...

$$C((X \times Y), (X \times Z)) = V(X) \times E(Y) \times E(Z)$$

Finally, in the APL solution later, we will actually use $V(X+Y)$ when Y is something like $J+(J \times P)$, and we will need two more formulae:

$$\begin{aligned} C(X, Y+Z) &= C(X, Y) + C(X, Z) \\ C(X, Y-Z) &= C(X, Y) - C(X, Z) \end{aligned}$$

Armed with all these expressions, it is possible to find $E(C)$ and $V(C)$ in any formula like the example. It could be very complicated, provided it only uses addition, subtraction and multiplication. By carefully breaking it down into individual additions, subtractions and multiplications we can eventually arrive at the result. Actually, $E(C)$ is always found simply by substituting the expectation of every variable in the model. Thus, in the example,

$$E(C) = E(J) + (E(J) \times E(P)) + E(B) + (E(K) \times E(L)) - (E(K) \times E(L) \times E(Q))$$

However, $V(C)$ quickly becomes a very messy formula indeed.

I needed to do this not just once but often. I had different models for costs in various parts of the task I was handling, and these were all subjected to modification in the course of the work. Furthermore, the expectations and variances of variables within each formula were changing, requiring the resulting formulae for $E(C)$ and $V(C)$ to be applied to new values each time. There were many other complications:

- the models were typically much longer than the example above;
- all the random variables were actually random vectors, which in particular meant that variances were actually variance-covariance matrices, with perhaps over a hundred rows and columns;
- means and variances of variables in the formula were specified in various different ways.

We ignore all of these complications hereafter, but they must be mentioned to explain the form of solution that I chose.

I decided that I needed an APL system that would accept just a symbolic definition of the model as a character vector, and do all the work for me. First, it would analyse the model to produce new character vectors, one for $E(C)$ and one for $V(C)$, in the form of APL expressions which could simply be executed to compute values for $E(C)$ and $V(C)$. It was clear that analysing the model to apply the above formulae would be a highly recursive process.

The Basic Solution

The example model becomes the APL character vector

```
MODEL+' +J+JP+B+KL-KLQ'
```

The rules are that the *MODEL* is a series of sections each beginning with a plus or minus sign, then one or more variable names. A further complication in the real problem (which changed the details of the coding quite substantially) was that variable names could be more than one character long. Notice that the model is understood in the usual algebraic sense, *not* as an APL expression. A minus sign, for instance, applies only to the product of variables it precedes. The rules were framed in this way because I did not want to allow parentheses. To have done so would have introduced numerous extra complications.

A function *EGET* was written to take as argument a model and to return a character vector with the APL expression for $E(C)$. This is a simple recursive function, the only complication being the need to add parentheses so that when the result is executed by APL the operations are done in the right order:

▽ $E \leftarrow EGET\ M;S;Y;Z$

[1] *a* Returns expectation of model *M*

[2] *SYZ*

[3] $E \leftarrow '(,((S='-')/S),(EPROD\ Y),)'$

[4] $\rightarrow(0=\rho Z)/0$

[5] $E \leftarrow E, EGET\ Z$

▽

▽ $E \leftarrow EPROD\ P$

[1] *a* Returns expectation of product *P*

[2] $\rightarrow(1<\rho P)/L$

[3] $E \leftarrow 'E', P$

[4] $\rightarrow 0$

[5] $L: E \leftarrow 'E', (1+P), ' \times ', EPROD\ 1+P$

▽

For instance, the result of *EGET MODEL*, with *MODEL* defined as above, is the character vector:

$+(EJ)+(EJ \times EP)+(EK \times EL)+(-EK \times EL \times EQ)$

There are surplus parentheses here, but it is better to have some that are not needed than to leave out any that are! The variables *EA*, *EB*, etc. will be in the workspace, containing the expectations of the variables in the model. Similarly, their variances will be in variables *VA*, *VB*, etc.

A More Complex and Interesting Recursion

Computing $V(C)$ requires more complex and interesting recursion, so I have given a listing of the key functions as an appendix. I will describe their operation in a bottom-up way, since this should be clearer than the mainly top-down way in which they were originally written. The basic idea of the recursion is to split the model into three parts:

- (a) the initial sign, '+' or '-';

- (b) the sequence of one or more letters following the sign, called a *product*, and
- (c) the rest of the model, which itself is a *model*.

There is a further recursion on splitting a product into its first letter and the remainder, which is another product.

Function *VPROD* takes as its argument a product and returns an expression for its variance, using the formula for $V(X \times Y)$. It first checks whether its argument is a single character. If so, it returns that character preceded by 'V'. Otherwise it applies the formula with *X* set to the first letter and *Y* to the rest. Since *Y* is in general a product, it needs to call itself for $V(Y)$. A simple recursion results. Notice that a local variable *YV* is defined as the result of *VPROD Y* and used elsewhere in the expression generated by the last line. The variable *YV* is created when we execute the character result of *VPROD*. Because of the recursion, executing the result of applying *VPROD* to a product of three or more variables means *YV* will be redefined during execution of the result. For instance *VPROD 'ABD'* gives the result:

$$(VA \times (EB \times ED) \times 2) + (YV \times EA \times 2) + VA \times YV + (VB \times (ED) \times 2) + (YV \times EB \times 2) + VB \times YV + VD$$

The assignment of *YV* first to *VD* then to a much more complicated expression causes no problems in execution, but does not make the expression easy to read. In general, the APL expressions resulting from even quite simple models in my system are quite unreadable - but they work!

The only other thing to notice about *VPROD* is the call to *EGET*. Since the argument of *EGET* is a model, which must begin with a sign, a plus sign is added, then removed from the result.

Next, the function *CPF* applies the formula for $C((X \times Y), (X \times Z))$. Its arguments are two products. It first calls a function *XYZ* whose role is to identify *X*, *Y* and *Z*. It sets *X* to the vector of letters common to *P1* and *P2*. Then *Y* and *Z* are the results of removing those letters from *P1* and *P2* respectively. *VPP* is not recursive. The only complication is the need to handle the cases when any of *X*, *Y* or *Z* is empty.

The next function up the system is *CPM*, which calculates $C(X, M)$ when *X* is a product and *M* is a model. It first calls a function *SYZ* to decompose *M* into sign (*S*), product (*Y*) and remaining model (*Z*), and if the last is empty just calls *CPF*. Otherwise it applies the formulae for $C(X, Y+Z)$ and $C(X, Y-Z)$. It calls itself recursively when it needs $C(X, Z)$. Finally, the same recursion is applied in the

top level function, *VGET*, which calls *VPROD*, *CPM* and itself in computing $V(C)$ by the formulae for $V(X+Y)$ and $V(X-Y)$.

The introduction of these few simple functions leads to high levels of recursion, even in quite short models. The expression for $V(C)$ generated by *VGET* for our example model has 197 characters.

Comments

Symbolic computation, or 'computer algebra', is currently attracting considerable interest. Ordinary computation puts numerical values into a formula and produces the numerical result. Symbolic computation manipulates the formulae themselves, using specified algebraic rules, to produce new formulae. There exist an increasing number of symbolic computation packages, and this is another growing field of computing activity with which APL needs to compete. This article has given just a simple application of symbolic computation in a particular specialised context. It does, however, show that APL is quite capable of handling such problems elegantly, and with little programming effort. APL's treatment of character data and its recursion capability are particularly useful.

For me, the exercise had many benefits. First, it saved me a lot of time. For the first few models I had to deal with, I worked out expressions for $E(C)$ and $V(C)$ by hand and programmed each resulting formula separately. Automating this process soon repaid the development effort. It had a second, unexpected, benefit when I was checking its results against my earlier hand-programmed models - I found two errors in my earlier work. I hate to think how many other errors I would have had to track down if I had not changed to an automatic system. The third benefit was in enjoyment. It is a lot more fun to work on something new than to do old things repetitively.

The one drawback was in speed. My hand-programmed solutions ran noticeably faster than executing the expressions generated by *EGET* and *VGET*. I made two modifications to improve execution speed, the first of which was to write a simple function to remove most of the redundant parentheses and plus signs from the expressions. The second was a compromise over allowing parentheses in the model itself. I allowed 'local' variables within the model to stand for submodels, provided that all 'local' variables remained independent of each other and of the remaining original variables. These two tricks enabled me to reduce the length of each expression considerably and so greatly increase execution speed. Symbolic computation will never produce code that runs faster than a good hand-programmed solution, but in the end I was very satisfied with my system.

Warwick University is participating in international research in symbolic computation, as well as having an ongoing interest in the use of APL in Statistics. I would be very interested to hear of anyone else who has used APL for symbolic computing, in however simple a form.

Appendix (remaining function listings)

▽ *V*+*VPROD P*; *X*; *Y*

- [1] *a* Returns variance of product *P*
- [2] $\rightarrow (1 \leftarrow \rho P) / L$
- [3] $V \leftarrow V', P$
- [4] $\rightarrow 0$
- [5] $L: X \leftarrow 1 \leftarrow P$
- [6] $Y \leftarrow 1 \leftarrow P$
- [7] $V \leftarrow (V', X, ' \times ', (1 \leftarrow EGET$
 $' + ', Y), ' * 2) + (YV \times E', X, ' * 2) + V', X, ' \times YV \leftarrow ', VPROD Y$
- ▽

▽ *C*+*P1 CPP P2*; *X*; *Y*; *Z*

- [1] *a* Returns covariance of products *P1* and *P2*
- [2] XYZ
- [3] $C \leftarrow ' '$
- [4] $\rightarrow (0 = \rho X) / 0$
- [5] $C \leftarrow VPROD X$
- [6] $\rightarrow (0 = \rho Y) / L$
- [7] $C \leftarrow (1 \leftarrow EGET ' + ', Y), ' \times ', C$
- [8] $L: \rightarrow (0 = \rho Z) / 0$
- [9] $C \leftarrow (1 \leftarrow EGET ' + ', Z), ' \times ', C$
- ▽

▽ C+P CPM M;S;Y;Z;L1;C1;L2;C2

- [1] *a* Returns covariance of product P and model M
- [2] SYZ
- [3] →(0=ρZ)/L
- [4] L1+0<ρC1+P CPP Y
- [5] L2+0<ρC2+P CPM Z
- [6] C+((L1^L2)/'('),(L1/S,C1),((L1^L2)/'+)'),L2/C2
- [7] →0
- [8] L:L1+0<ρC1+P CPP Y
- [9] C+L1/S,C1
- ▽

▽ V+VGET M;S;Y;Z;L1;C1

- [1] *a* Returns variance of model M
- [2] SYZ
- [3] →(0=ρZ)/L
- [4] L1+0<ρC1+Y CPM Z
- [5] V+'(',(VPROD Y),'')+(',(VGET Z),''),L1/'+2×',S,C1
- [6] →0
- [7] L:V+VPROD Y
- ▽

▽ SYZ;N

- [1] *a* Splits model M into sign S, product Y and model Z
- [2] S+M[1]
- [3] Y+1+M
- [4] N+~1+(Y1+'+')[Y1:'-'
- [5] Z+N+Y
- [6] Y+N+Y
- ▽

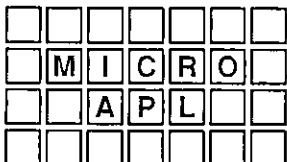
▽ XYZ

- [1] *a* Identifies common terms X in P1 and P2,
- [2] *a* setting Y and Z to the residues
- [3] I+P1:P2
- [4] X+(I≤ρP1)/P2
- [5] Y+P1[(~P1<X)/!ρP1]
- [6] Z+(I>ρP1)/P2
- ▽

MicroAPL for ...

APL and networking consultancy

MicroAPL has 10 year's experience with small-computer APL systems. We have pioneered the usage of networked microcomputers and can plan, specify, install and maintain PC networks of all sizes. We have experience with both Ethernet and Token-ring networks and their mainframe gateways. Our programming skills cover all areas of APL and systems programming. MicroAPL can also train and supervise your APL staff on your behalf. If you would like to review your systems requirements, contact David Eastwood at:



MicroAPL Ltd.
South Bank Technopark
LONDON, SE1 6LN
Telephone: 01 922 8866

MicroAPL for ...

APL on the widest range of micros

STSC's APL*PLUS Release 8	£425
STSC's APL*PLUS II PC	£1395

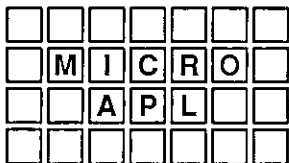
APL.68000

APL.68000 for the Mac, ST, Amiga	£86.91
APL.68000 for the Mac II, SE30	£260.82

APL.68000 runs on many multi-user computers, including:

Sun, NCR, Wicat, HP200/300, Spectrum, Aurora, Stride £1200 to £2000

All prices quoted exclude VAT



MicroAPL Ltd.
South Bank Technopark
LONDON, SE1 6LN
Telephone: 01 922 8866

Access and Barclaycard
Orders accepted

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

Hackers' Corner	Adrian Smith	90
Technical Correspondence	Peter Branson et al	91
Error Trapping in IPSA APL	Peter Biddlecombe	102
Problems for APL Buffs	Zdenek Jizba	109
Using the GEM File Selector	John Sullivan	113
A Vector Construction Kit	Adrian Smith	117

Hacker's Corner: Beware the M61

by Adrian Smith

**** Urgent Warning ** ** Urgent Warning ** ** Urgent Warning ** **
** ** Urgent Warning ** ** ** Urgent Warning ** ** ** Urgent Warning ****

Should you be contemplating the purchase of an IBM PS/2 model-70 M61 to run your APL*PLUS/PC systems don't! Rowntrees have just taken delivery of 5 of these machines, and so far none of them will reliably get beyond *CLEAR WS*. I think that the problem is rooted in the new half-size motherboards, my evidence for this conjecture being:

- I use the (now obsolete) 16 MHz F61, and have had a happy and trouble-free time. My colleague took delivery of an ostensibly identical machine, which gave the weird symptoms outlined below. Accordingly we called in IBM to fix the machine, and here the fun began. The engineer attempted to swap the motherboard for the one he had brought with him, and found to his astonishment that it wouldn't fit, as the F61 had a strange half-size board he had never seen before. He checked that his (full-size) board was the correct one, and did a bit of nifty rewiring of the speaker leads to make it fit. The machine has run trouble-free ever since.
- IBM then quietly withdrew the F61, and are shipping the 20MHz M61 in its place. The very first of these showed the same symptoms as the above-mentioned F61, and on removing the case we observed the same half-size board! We already had several of these on order, and sure enough, they all behave in exactly the same way!

The effect occurs at random, but seems to be easier to produce on a fully warmed up machine. Essentially, APL completely loses track of the keyboard, such that (for example) the left shift key gives an \supset , the escape key gives F1, and the Prt-Scrn key reboots! Obviously, a power-off reset is the only escape from this state! On two occasions, APL continued to work normally, but on exit to DOS the symptoms manifested themselves as described above.

I am told that IBM have accepted that there is a problem with one of the EPROMs on the motherboard, but I have not yet (27th Jan) heard of a date when a replacement is likely to be shipped.

TECHNICAL CORRESPONDENCE

Is it APL2, or is it me?

From: Peter Branson

December 1989

A while back (Vector 4.2.99), I enlarged Adrian Smith's $\Delta STRIP$ function to do a stand-alone search for global (or semi-global) names in individual functions, under VSAPL. Recently I moved to a new site with APL2 (Release 2), and since there was nothing comparable immediately available, I put this function up again (using the better 'del-all-dupes' idiom from Vector 4.4.97, of course).

Unlike VSAPL, APL2 allows trailing comments on code lines, so I put in some (APL2-type) code to handle these in the same position as before, plus some other minor improvements. At first it seemed OK, and worked on some small functions, but then I tried it on itself, with this result:

```

      STRIPX ' $\Delta STRIPX$ '
FAIL
NAME
0123456789
 $\Delta TAB$ 

```

and only ΔTAB should be there!

I double-checked for typos and possible name coalescence, etc., but could find nothing wrong. Finally I narrowed the problem down to the 'delete if in quotes' line:-

```
V←(¬(V='''')∨≠\V='''')/V
```

I have looked at this idiom in all sorts of ways, including breaking it right down to the individual boolean results, and it seems alright on various test objects. It is difficult for me to do a detailed analysis on ' $\Delta STRIPX$ ' itself since I cannot print

anything yet, and the intermediate results are a bit large to analyse from screen displays in the limited time available, so I tried a different approach.

Since the trouble occurs with quoted bits and a label name well down in the `□CR`, I created a little *DIQ* (Delete in Quotes) function, which just does the above job, plus delete multiple blank rows for a neater display. The matrix which this works on was caught from `ΔSTRIPX` into a global variable, and *DIQ* was run against this with progressively fewer of the top lines dropped off, i.e. *DIQ*, `(N, 0) + MAT, ' '` with reducing *N* (taking care not to drop the lines causing the trouble).

The results were surprising. The function has 27 lines (different from the original because of trailing comments, etc.) and for values of *N* of 12 and above *DIQ* is fine; the result is a vector occupying a few screen lines holding bits of function lines, all entirely as expected with no rogue entries at all. At *N* = 11 there is an abrupt change, the result of *DIQ* now being:-

```
□WA □FX : 0123456789 INVALID FN NAME
```

and that is all there is! Thereafter, as *N* is progressively reduced to 0, the result gradually builds back upwards from this short result, but now always containing rogue bits.

The matrix, *MAT*, is formed in this `ΔSTRIPX` by disclosing the result of a localised sub-function and checks OK at depth 1. However, when IBM's DISPLAY function is applied to it, two of the right hand vertical border lines are displaced one print position to the right; I have not seen this before, but the positions don't have any obvious relation to the problem lines.

At present I am baffled. Unless I am missing something obvious (or perhaps quite subtle), it's weird. I don't yet have any access to IBM so???????

Peter Branson

More on Ambivalent Operators

From: Dave Piper

December 1989

Following my letter to Vector discussing an *AMBIVALENT* operator to aid the implementation of ambivalent functions, I received the enclosed letter from Claude Henriod in Switzerland. Subsequently I sent the following reply:

Dear Claude,

Thanks for the feedback on the ambivalent operator. I apologise for the error omitting a pair of parentheses - they got lost somewhere between keyboard and letter! The error you found occurs because operators have short scope to the right, but long scope to the left. Thus `□NC` was taken as the right operand and the quoted string as the right argument to the derived function.

I found your timings most interesting, especially the second batch where the ambivalent operator appeared to provide the quickest solution. I guess this could be a quirk in the implementation. I am opposed to the use of `execute` to achieve the same result since you lose control of the code (the strength of the function is reduced). In the branching solution you provide, I strongly prefer to branch to a label rather than `□LC+x`. I would also use the skipped line to assign a default left argument (as in the `execute` version) rather than perform a modified calculation.

The functions *AVERAG2* and *AVERAG3* are coded less efficiently since you perform the division before the plus-reduction. This means you are performing many divisions instead of just one. I suspect this could have produced the timing results noted above.

My own criticism of the operator I proposed is that it is still much too complex to achieve such a simple effect. That does not imply I can see a simple solution, however.

I am forwarding a copy of your letter, listings and timings to VECTOR for publication, along with a copy of this letter.

I should like to thank my wife, Helen, for providing a translation of the above into French.

Yours faithfully,

D B Piper,
41 Sandown Drive,
Rainham,
Kent ME8 9DT

From: Claude Henriod

21 July 1989

Concern: a possible feedback.

Dear David,

I have just received Vector Vol 5 no 4 with your technical information about an "Ambivalent Operator". I found them very attractive for the comprehension and usage of a programmed operator. On my own usage I prefer the Execute form which is more closely to an idiom of APL. Next, the monadic *AVERAGE* produces a recursive loop taking more time. The line *AVERAGE[2]* is not very easy to understand and more complicated than the Execute or a 3 line processing. With I-APL using "Direct Definition" (stmt of maximum one line with 3 expressions) the recursive principle is of great interest.

FUNC: L-Exp. (alpha, omega) : logical-Exp : default FUNC omega

```

)WSID
MPCALC
  * MULTI-PRECESSION CALCULATOR IS A WS OUT OF THE OFFERED
DISK
  * TO I-APL BY C.H.I. Swiss

  * ISUB IS A SAMPLE OF A RECURSIVE FUNCTION IF MONADIC CALL
  * THIS FORM IS VERY ATTRACTIVE FOR DIRECT DEFINITION

ISUB:
ISUB: ICAN-/α ISYN ω : 2*[NC 'α' : 0 0 ISUB ω

    0 1 2 ISUB 0 5
0 9997
    ISUB 0 5
0 -5

)CLEAR
CLEAR WS
)IN 10 PIPER
SAVED 1989-07-21 08.16.52 10 PIPER
)FNS
AVERAG0 AVERAG1 AVERAG2 AVERAG3 XTIME
)VARS
MINITESTDY MINITESTMO MINIVEC TESTDY TESTMO VEC
)OPS
ΔMONADIC
  (ρVEC) (ρMINIVEC)
20 5

```



```

V AVERAGO[[]]V
[0] AVE+SIZE AVERAGO NUMS;RUN
[1]  A FUNCTION LIKE D.B. PIPER, USING AN OPERATOR
[2]  +(+ (RUN AVE)+(+ (P NUMS)(AVERAG1 ΔMONADIC [NC 'SIZE'] NUMS))/O
[3]  AVE+(SIZE+/NUMS)÷SIZE
V

```

```

V AVERAG1[[]]V
[0] AVE+SIZE AVERAG1 NUMS;RUN
[1]  A FUNCTION LIKE D.B. PIPER, USING AN OPERATOR
[2]  +(+ (RUN AVE)+(+ (P NUMS)(AVERAG1 ΔMONADIC ([NC
'SIZE']) NUMS))/O
[3]  AVE+(SIZE+/NUMS)÷SIZE
V

```

```

V AVERAG2[[]]V
[0] AVE+SIZE AVERAG2 NUMS
[1]  A FUNCTION USING EXECUTE FOR SIMPLE VISUALISATION
[2]  ±(2* [NC 'SIZE'] )/'SIZE'+P NUMS'
[3]  AVE+SIZE+/NUMS÷SIZE
V

```

```

V AVERAG3[[]]V
[0] AVE+SIZE AVERAG3 NUMS
[1]  A FUNCTION, WITH OR WITHOUT LEFT ARGUMENT
[2]  +((2=[NC 'SIZE'])/[LC+2
[3]  →O AVE++/NUMS÷P NUMS
[4]  AVE+SIZE+/NUMS÷SIZE
V

```

A HERE, IN THE MONADIC CASE, THE RESULT IS A SCALAR.

A NOTE: AVERAGO DOESN'T WORK WITH APL2/PS2
 AVERGO VEC

SYNTAX ERROR

ΔMONADIC[2]

```

[2] +((2=CLASS)/R+O
    ^

```

[NC 'CLASS']

3

A TIMINGS WITH APL2/PC

```

<XTIME"TESTMO
AVG CPU: 0.0291 Sec for 100 times of: R1+AVERAG1 VEC
AVG CPU: 0.0171 Sec for 100 times of: R2+AVERAG2 VEC
AVG CPU: 0.0148 Sec for 100 times of: R3+AVERAG3 VEC
<XTIME"TESTDY
AVG CPU: 0.0264 Sec for 100 times of: R1+15 AVERAG1 VEC
AVG CPU: 0.0286 Sec for 100 times of: R2+15 AVERAG2 VEC
AVG CPU: 0.0291 Sec for 100 times of: R3+15 AVERAG3 VEC
<XTIME"MINITESTMO
AVG CPU: 0.0291 Sec for 100 times of: R1+AVERAG1 MINIVEC
AVG CPU: 0.0154 Sec for 100 times of: R2+AVERAG2 MINIVEC
AVG CPU: 0.0142 Sec for 100 times of: R3+AVERAG3 MINIVEC
<XTIME"MINITESTDY
AVG CPU: 0.0203 Sec for 100 times of: R1+4 AVERAG1 MINIVEC
AVG CPU: 0.0159 Sec for 100 times of: R2+4 AVERAG2 MINIVEC
AVG CPU: 0.0148 Sec for 100 times of: R3+4 AVERAG3 MINIVEC

```

A TIMINGS WITH APL232

```

<WSTIME"TESTMO,TESTDY,MINITESTMO,MINITESTDY
AVG CPU: 0.0235 Sec for 100 times of: R1+AVERAG1 VEC
AVG CPU: 0.0122 Sec for 100 times of: R2+AVERAG2 VEC
AVG CPU: 0.0085 Sec for 100 times of: R3+AVERAG3 VEC
AVG CPU: 0.0203 Sec for 100 times of: R1+15 AVERAG1 VEC
AVG CPU: 0.0149 Sec for 100 times of: R2+15 AVERAG2 VEC
AVG CPU: 0.0165 Sec for 100 times of: R3+15 AVERAG3 VEC
AVG CPU: 0.0205 Sec for 100 times of: R1+AVERAG1 MINIVEC
AVG CPU: 0.0109 Sec for 100 times of: R2+AVERAG2 MINIVEC
AVG CPU: 0.0099 Sec for 100 times of: R3+AVERAG3 MINIVEC
AVG CPU: 0.0154 Sec for 100 times of: R1+4 AVERAG1 MINIVEC
AVG CPU: 0.0138 Sec for 100 times of: R2+4 AVERAG2 MINIVEC
AVG CPU: 0.0105 Sec for 100 times of: R3+4 AVERAG3 MINIVEC

```

A WSTIME EXECUTES A □WA BEFORE EACH ELEMENTARY LOOP

□WA
5166576

Best regards

Claude Henriod,
Monteiller,
CH - 1965 Savièse,
Switzerland.

Yet More on that Ambivalent Operator ...

From: Norman Thomson

15th December 1989

My heart leapt up when I beheld the headline "Ambivalent Operators" in the Technical Correspondence section of Vector Vol.6 No.2, but sadly the promise of what I deem an ambivalent operator, that is one which can accommodate either one or two operands, was not fulfilled!

I feel that the use of lengthy descriptive names serves in the case of this correspondence to obscure rather than elucidate the underlying discussion which is nevertheless worth while following. Accordingly I shall attempt to clarify what I believe to be the salient points arising from David Piper's and Phil Last's contributions.

It all starts from the problem of providing an ambi-valent function which returns for a right argument R the moving average of period L if the left argument L is present, and the simple average if it is not, and also avoids the use of either branching on $2=\square NC 'L'$ or conditional \pm .

David's proposal is the following (I have amended his operator MON slightly so that the operand Q is a general condition - this seems to me to make it more useful than having a test for $Q=2$):

```
[0]  Z+L(P MON Q) R
[1]  +L1 IF Q
[2]  +0 Z+1(L P R)
[3]  L1:Z+0
```

```
[0]  X+L IF R
[1]  Z+R/L
```

```
[0]  Z+L AVP R;T
[1]  +0 IF+(T Z)+(pR)(AVP MON(2=square NC 'L'))R
[2]  Z+(L+R)/L
```

```
3 AVP :5
2 3 4
  AVP :5
3
```

The essence of the method is that the derived function *P MON* returns a flag which is 1 for dyadic, 0 for monadic, followed by the value in the former case. In the latter case it re-executes *P MON* with in the present example (ρR) as left argument.

As Phil points out it is somewhat unsatisfactory always to re-execute in the monadic case. Moreover a function applied as an operand to *MON* must be constructed in the rather special way that *AVP* is, with a default left argument supplied as in line 2. This goes against the general principle that operators should have as wide a range of applicability as possible.

Phil argues that there is no case for an operator and uses a function *AM* as shown below. In the event of *L* being absent a "do-nothing" function *L* is constructed. This means of course that the technique is only useful for functions like the present one where the monadic meaning is the same as the dyadic meaning with the left argument scored out. I suspect that the number of functions of any practical use where this applies is very limited.

```
[0]  AM R
[1]  R+□FX 'Z+L R' 'Z+R'
```

```
[0]  Z+L AVL R
[1]  AM 'L'
[2]  Z+(L+/R)÷+L, ρR
```

```
      3 AVL :5
2 3 4
      AVL :5
3
```

Phil's version of *AM* uses the same name for function and right argument - a permissible if potentially confusing use of the language. There may be some hidden advantage in using this here, but if so it escapes me. Another disadvantage of this technique is that the function call to *AM* must be explicitly built into all functions which use it in this way.

One way to combine David's and Phil's ideas might be to define an *AMBI* operator...

```
[0] Z+L(P AMBI)R
[1] □TS+□FX 'Z+L R' 'Z+R'
[2] Z+P R
```

```
[0] Z+AVX R
[1] Z+(L+/R÷+L, ρR
```

```
3 AVX AMBI 15
2 3 4
AVX AMBI 15
3
```

which works, but the trouble now as the observant reader will have noticed is that *L* does not appear in the function header for *AVX*, and so the construction of *AVX* presupposes its use as an operand of *AMBI*. If *L* is localised in *AVX*

```
[0] Z+L AVX R
```

it cannot pick up the value from the encompassing operator and so trouble ensues....

```
3 AVX AMBI 15
VALUE ERROR+
AVX[1] Z+(L+/R)÷+L, ρR
      ^^
```

The same sort of consideration applies to the simple-minded operator approach based on $2 = \square NC 'L'$ branching:

```
[0] Z+L(P AMB)R
[1] →L1 IF 2=□NC 'L'
[2] →0 Z+P R
[3] L1:Z+L P R
```

```
3 AVX AMB 15
2 3 4
AVX AMB 15
VALUE ERROR+
AVX[1] Z+(L+/R)÷+L, ρR
      ^^
```

Returning to the original problem, perhaps the conditional \pm solution is the best after all:

```
[0]  Z+L AVG R
[1]  Z+±('Z, '+/R)÷+' , (Z+'L' IF 2=□NC 'L') , 'pR'

      AVG 15
3
      3 AVG 15
2 3 4
```

Meanwhile the quest for the holy grail, alias the wholly ambivalent operator, goes on!

VCAT: A Response

While it gives me great pleasure to see Peter Branson's contribution to the Education Vector (Vol.6 No.2) I wonder whether his *VCAT* example reflects the ways in which we should be encouraging APL newcomers to tackle such a problem now that APL2 is becoming so commonplace that even kids can have it for nothing (providing of course that they have access to a computer made by a certain well-known manufacturer...!)

The modern didactic view of this situation is to make each of the two matrices into vectors of the rows ($\leftarrow [2]''$), join them ($, /$), then restore the second dimension by de-nesting (\Rightarrow).

Here is the function:

```
[0]  Z+L VCAT R
[1]  Z+⇒, /←[2]''L R

      LEFT RIGHT
BREAD MAN
FRUIT CAN
      EAT

      LEFT VCAT RIGHT
BREAD
FRUIT
MAN
CAN
EAT
```

The second disclose is necessary on account of the implicit enclosure arising from the derived function , /.

This *VCAT* is fine provided both arguments are matrices. If the function is to work with vectors or scalars, we have to generalise $c[2]$ to:

```
[0]  Z+EXPAND R
[1]  Z+c[ρρR]R+1/R
```

$1/R$ which makes R into a 1-item vector if it is a scalar, otherwise does nothing.

Rewriting *VCAT* we have:

```
[0]  Z+L VCAT R
[1]  Z+⇒, /EXPAND" L R
```

```
LEFT VCAT '3'
BREAD
FRUIT
3
3 VCAT 4
3
4
'PIG' VCAT 'SHEEP'
PIG
SHEEP
```

Norman Thomson, Mail Point 188,
IBM UK Laboratories,
Hursley Park, WINCHESTER,
Hants SO21 2JN.

Ed: this looked interesting enough to time! From the results below, I would have to say that I side with Peter, at least until someone comes up with an interpreter that will run Norman's code as fast for realistic sizes of data!

ρLEFT		ρRIGHT		Z+L VCAT R	
				NT VCAT	PB VCAT
2	5	3	3	270	380
12	80	23		1480	390
56	132	63		9500	450

A screen builder
A report builder

SHARP APL Event Trapping A Tutorial Introduction

by Peter Biddlecombe (I.P.Sharp Associates)

Abstract

This article describes the event trapping facilities available in Sharp APL. As the basic elements of event trapping were covered by the articles in VECTOR 6.1, they are not repeated here. Readers wanting to refresh their memory should use Pauline Brand's article on error trapping in Dyalog, as Sharp APL, like Dyalog, traps specific events rather than 'any error'. This article does not cover the use of error trapping in areas specific to Sharp APL such as batch and non-terminal tasks.

Errors and Events

In Sharp APL we trap events, not just errors. Events include interrupts and 'return to immediate execution'. When we trap an event, we simply specify an alternate set of actions to be performed when an event occurs. This is appropriate when

- (i) a particular event does not always happen, or may happen at any time; and
- (ii) we know the required action when it does happen.

Handling of Untrapped Events

If an untrapped event occurs, then APL execution halts, and an error message is displayed in the usual manner. The text of this error message is stored in the system variable $\square er$, which is normally a 3-row character matrix.

The rows of $\square er$ contain the event number and an event title, the function line or APL statement in which the error occurred, and the caret indicating the point reached by the APL interpreter.

Event Numbers

Event numbers are divided into the following four groups (The number ranges shown are those reserved for event groups, not the ones actually used):

1-499 Errors detected by the system

These include some events (e.g. *FILE SYSTEM NOT AVAILABLE*) which can't be fixed by actions taken inside the active workspace.

500-999 User-defined errors

System function `□signal` can be used to signal an error in the same fashion as an error detected by the system, and such errors can be trapped. Any event number from 1 to 999 can be used by `□signal`, but it is bad practice to use numbers below 501.

1000-1999 Interrupts

These are signals which cause execution to be halted, and are classified by the activity in progress (e.g. *INPUT INTERRUPT*), as well as the type of interrupt.

2001 Return to immediate execution

This event does not cause `□er` to be set, but can be trapped. The range of actions which can be carried out when it is detected is restricted. This event is often trapped as a security measure in systems where the user should not be using immediate execution mode.

Using `□signal` for User-defined Errors

Syntax: [error text] `□signal` event

where *error text* is optional, and *event* is treated like a right argument of 'branch' - if it is empty, no error is signalled, otherwise the first number is used, and the event is signalled in the same way as an error detected by the system, with the optional left argument *error text* included in `□er`. User-defined errors can be trapped.

The format of `□trap`

Each `□trap` setting can include any number of *trap* definitions. The data assigned to `□trap` may be a character matrix or character vector. The matrix

form contains one trap definition per line. The vector form (which is more commonly used) **must** contain a delimiter as its first character.

The Format of Trap Definitions

Each trap definition contains at least one part, and two optional others, separated by blanks:

- 1 Event number or numbers (optional) - a list of event numbers or ranges of numbers to which the definition applies. If omitted, a default set of event numbers (depending on action code) is used.
- 2 Action code (mandatory) - a single letter, optionally preceded by an action code modifier.
- 3 Trap line or keyword (optional for some actions). Tells the system what to do following the event, and how to resume execution if this is required. A trap line may use multiple statements separated by diamonds.

Treatment of 'bad' \square trap Settings

If a \square trap setting does not obey these rules (e.g. if the delimiter is omitted from the start of a vector trap setting), then it is set by the system to an empty vector - no error is signalled. Because trap lines are not used until the relevant event occurs, a trap line which causes an error will not come to light until the trap line is tested or the error occurs. Moral: test your trap lines!

How the System Decides Which \square trap to Use

When a trappable event occurs, the system searches the most local value of \square trap (from left to right) until it finds an action for the event. If it finds none, it continues searching in the next most local shadowed value, until either a value is found or the stack is exhausted. This search does not of itself affect the state indicator or $\square LC$. This treatment is **different** to that of other local variables, and means that:

1. As long as a trap definition exists for a particular event, it applies unless a more local definition supersedes it.
2. We **cannot** suspend event trapping by setting a localised \square trap to an empty vector, but in any particular function we can replace the standard trap definitions by using trap definitions with event ranges covering all events.

Possible Actions in Trap Definitions

There are six possible action codes and one action code modifier, which are used in definitions as follows:

Format		Name
[evnums]	[O] C [trap line]	Cut
[evnums]	[O] E [trap line]	Execute
[evnums]	[O] N	Next
[evnums]	[O] S	Stop
[evnums]	[O] D [keyword]	Do
6	[O] I	execute In mid-line

Where *evnums* is a an event number list or range.

C - Cut back the stack and execute trap line

If an event in *evnums* is detected, all functions called later than the one in which this definition is localised are removed from the SI stack. The state indicator and *LC* are amended, and *er* is set. Note that any line label included in the trap definition is interpreted in the context of the function where the definition is localised, **not** the function where the error is detected. Trap settings which use action code C should be localised.

E - Execute trap line

If an event in *evnums* is detected, the system immediately sets *er*, and then executes the trap line, acting as if execution had been halted and the trap line entered at the terminal. This means that any line labels in this setting are interpreted in the context of the function in which the error occurred, not necessarily the one containing the trap definition. This means that the only branch whose effect is certain is to *ic*.

Example - using a trap definition in a 'main' function to handle errors which are not trapped at a more local stack level:

```
[trap+ 'v 900 C +errline v 0 e errlog o [signal 900'
```

(*errlog* is a function which files details of the error, *errline* is a line label in this function. Event number 0 denotes the range 1-999.)

N - Skip to next level

If an error in evnums is detected, the system abandons the search for trap definitions at this level of localisation, and moves to the next one where `□trap` is localised. It can be used just before a trap definition which uses a range of event numbers, in order to remove a few event numbers from the range.

S- Stop

When an event in evnums is detected, the system halts work and displays `□er`. This action code is mainly used in debugging to suspend the trapping of certain events.

D- Do keyword

This is the only action which may be used with event 2001 (return to immediate execution). If no keyword is used, no action is taken. Valid keywords are:

`EXIT` signals event 2001 to the next level of localisation, where event 2001 may be trapped in turn.

`CLEAR` clears the active WS.

`OFF` does an automatic `)OFF`.

I- execute In mid-line

This code can only be used with event 6 (`VALUE ERROR`). It is similar to E except that (i) information about the execution of the current line is retained, (ii) if no branch is specified, execution is resumed at the same point as the error occurred, and (iii) an additional line is catenated onto `□er`, containing the name of the missing object.

Example - use of a global trap definition to trigger a function which fetches objects from file if they are not in the WS when required:

```
□trap+ '∇ 6 i fetch □er[□10+3;]'
```

O- Only (Action code modifier)

This restricts a trap definition to the environment of the function in which it is set. This includes uses of `□` but not calls to other functions.

Watch out!

Event trapping is a double-edged sword, and if misused can cause as many problems as it solves. Some reasons for this are:

1. It provides a method of branching which is not explicit in all the functions which could be affected.
2. A trap line containing an error may not be executed for many months or years after it is written.
3. Badly formed traps are ignored without comment.
4. It is possible to write trap lines which cause endless loops.
5. The system can see trap definitions which a programmer can't. In a suspended function, `□trap` entered at the terminal will only show definitions at the current stack level.

Anyone about to embark on using `□trap` is well advised to read the advice in Chapter 13 of the Sharp APL reference manual on testing functions which use event trapping.

□ec - Environment Condition

This system variable exists to add an extra level of security in systems where it is desired to prevent users from interrupting processing. Although `□trap` can be used in the first line of a function to trap interrupts, it is possible to generate an interrupt before `□trap` has been assigned, and thus suspend the function.

□ec may be set to 0 (the global default) or 1. Whenever it is localised in a function, the system immediately sets it to 1 when the function starts execution. If any error occurs when □ec is 1, the SI stack is cut back one level, and the error is signalled again. If □ec is 1 at any level of the stack, this process is repeated until □ec is zero at all stack levels.

ATTENTION (event 1002) is ignored, but left pending and responded to as soon as □ec is zero at all SI levels. Other Interrupt events are treated like errors. Return to immediate execution and stops via `sΔ` are treated normally, which allows us to use stop vectors for debugging purposes without removing assignments of □ec.

How `vec` Works when Event Traps are Set

When an event occurs, each level of the SI stack is examined, starting at the most local level. The following tests are made, and if neither is satisfied at any level, the default action for handling events is taken.

- (1) Does the visible `trap` contain a definition for the event? If so, and the visible trap definition is localised at this level, execute the trap line.
- (2) Is `vec` localised at this level? If so and it is set to 1, handle the event as in (1). If it is 0, continue the search at the next level.

Summary

Event trapping is a very useful facility, provided it is used correctly. In combination with `vec`, it can be used to make systems secure, and to ensure that users are not required to type APL expressions or system commands.

Problems for APL Buffs

by Zdenek V Jizba

Author's Note

Last night, it occurred to me that some of the problems I coped with during my professional career might be of interest to the APL community. This morning I put together a few notes on one of these subjects, and am sending them to you under the title "Problems for APL Buffs"

The concept of a random number is closely associated with a population frequency distribution. When a random sample is taken, a histogram of this sample should look like the frequency function of the population from which the sample was taken. The larger the sample, the closer should the histogram approach in shape that frequency curve.

In APL, a source of random numbers is available with the roll:

```

      ?10p3
1 2 3 3 1 3 2 2 1 3

```

In the above example (I am using 1 as the index origin) the digits 1 2 and 3 should appear with roughly equal frequency. Therefore the underlying frequency distribution is a discrete function, and is defined as one third at the points 1 2 and 3. With the roll, and with some simple transformations, it is possible to generate random samples from arbitrary frequency distributions. A frequency function where each value has an equal chance of occurrence is called a *uniform distribution*.

Consider now a two-dimensional frequency function:

```

      ?3 2p10
5  3
1 10
1  6

```

Each row is now a sample of two values. The associated frequency function is not a curve (or more accurately, points on a line for the discrete case), but a surface. There are two dimensions for the random variables, and one dimension for the probabilities. The extension to higher dimensions is straightforward.

Rotations

Another type of random sample is one dealing with rotations. Consider points in the x/y plane:

```

P+3 2p5 0 4 3 3 4
      P
5 0
4 3
3 4

```

If we desire to rotate these points 'randomly', what does that mean? (From here on, whenever I use the word *random* I will mean 'random in a uniform sense'.) The analogue of a uniform distribution for a rotation is one where each angle from 0 to 360 is equally likely. But given the angle, one still has to come up with a transformation that would change X,Y values to a new set of rotated ones X',Y'. We can readily define such a function:

```

V U+ROT W
[1]  a Two dimensional rotation
[2]  U+2 2 p 1 1 -1 1 x 2 1 1 200W+180
V

```

Suppose that we desire to rotate the rows in *P* in such a way that the first point (row-1 of *P*) moves to the place of the second point. (The angle is that for which the tangent is 3÷4 or 36.8698976...)

```

Q+P+.xROT M+36.86989764584402
      Q
4          3
1.4      4.8
4.4408921E-16 5

```

To rotate these points back, there are two ways of doing it:

```

Q+.xROT -M
5          4.4408921E-16
4          3
3          4

```

or ...


```

      Q+. *ROT M
5      4.4408921E-16
4      3
3      4

```

The reason is that the two-by-two array is a special one, belonging to a class of matrices called *unitary*. Its transpose is equal to its inverse; the matrix multiplied by its transpose produces the unit matrix with ones on the diagonal and zeros elsewhere.

Moving on the Sphere

The next question that one might ask is: what about three dimensions? Suppose we have points in space and we want to rotate these in a random way: how could we define *random*?

Things get a little more complicated here. To 'simplify' matters, consider a landmass on the globe (we will assume here that the Earth is a perfect sphere). If we desire to move England to, say Australia, we could do that in two steps. First we move the centre of gravity of England to the desired point. Then we rotate England around this centre of gravity as in the two dimensional case. Intuitively, a *random* motion on the unit sphere would be defined by picking any point on the sphere with equal likelihood, followed by a random rotation.

How would one define an APL function that randomly selects a point on a sphere? THAT might be a useful function. But is the motion that we have just defined truly random? Consider the following: it is well known that any motion on a sphere is equivalent to a simple rotation. The pole of that rotation has a special name: it is called the Euler pole. Given an Euler pole, and an angle of rotation, one can generate any motion of a two-dimensional object on the surface of a sphere.

This provides us with a second possible definition of random motion on a sphere: first select a random location on the sphere as before; then call this point an Euler pole, and select a random rotation in the plane. Are these two definitions of random motion equivalent? (I do not know the answer!)

So we have several interesting APL problems here. First consider writing an APL function that would generate a random unitary matrix for rotation in three dimensions. Next, given such a matrix, find the associated Euler pole and rotation. Another problem: given two points on the surface of a sphere (say London and Edinburgh) and a new location for the first point and a geographic direction for the second one (i.e. a desired rotation) find the appropriate unitary

matrix. Finally: given an Euler pole, and a rotation, find the associated unitary matrix.

If these problems are too trivial, try this one. Define random rotation in more than three dimensions, and write an algorithm to generate the associated unitary matrix. Make sure that the matrix involves **only** rotations, and does not include inversions, which is a totally different subject.

If these problems are too abstract, here is one that is more practical: you have a map on the scale of 1:25000 from very high latitudes (the grid of parallels and meridians is not cartesian). You can digitize points on that map to about 4 significant digits of accuracy. Write an APL function to convert the digitized cartesian co-ordinates to latitudes and longitudes. If you use some of the functions discussed above, your function should be less than six lines of code.

How to Use the GEM file Selector

by John Sullivan

An Interface Program for APL.68000 on the Atari ST

The GEM file selector is a program built into GEM which allows application programs to present the user with a dialogue resulting in the selection of a file, the path and name of which are returned to the application program for further processing. Anybody who has used any serious Atari software (for instance the word-processing program that usually comes bundled with the machine) will know how useful the file selector is (and also how terrible the Atari one can be!). It would be most useful if this facility were available from APL, but APL.68000 does not have a built-in interface for it.

There is a way around this apparent omission: write your own interface. This is not difficult if you are an assembler programmer, because MicroAPL have thoroughly provided a way of doing just that, by means of an auxiliary processor [1]. However, not everybody is an assembler programmer, hence this article. (You may say, after reading my assembler code, that I'm not an assembler programmer either. It's a fair cop - I admit it!)

How to use the function

The cover function is called <FSEL>. It takes character left and right arguments and returns an explicit result. This result is the full name of the selected file including the path, unless 'cancel' was specified, or if an error occurred in the AES call, in which case the null vector is returned. If an APL error occurs (such as WS FULL or NONCE ERROR) the function will stop in the usual manner: this can be trapped and dealt with by the function that calls <FSEL>. The only 'subroutine' function is <TRANSLATE> from the TOOLS workspace (supplied with APL.68000) which is used to convert the character strings from ASCII to the internal format used by APL.68000 and vice versa.

When using the file selector the idea is that the options that were selected last time are used as the starting point for next time. These options can, of course, be over-ridden by appropriate use of the left and right arguments to the cover function <FSEL>. However, in order to save preferences from one call to the next, the assembler code is held in a global variable <FSELmc>, rather than being initialised by the cover function every time it is called, which is the normal way

of dealing with auxiliary processors. Of course, you can re-initialise the code whenever you like by calling the setup function: an ideal time to do this is on entry to the workspace, in the latent function. It is incumbent on the application programmer to ensure that the global variable does not become corrupted in any way.

The left argument to the function is the initial path name to be used. If this is omitted or null, the program uses the path that was returned last time the file selector was used, or 'A:*.*' if it hasn't been used since initialisation. This path can be up to 63 characters long, which is the maximum path length in DOS (but be careful, because there are a few peculiarities with path lengths on the Atari).

The right argument to the function is the initial filename to be used. If this is null, the program uses the filename that was returned last time the file selector was used, or it displays a blank filename if it hasn't been used since initialisation. The maximum filename length is 12 characters with an optional embedded period which, if required or used, must be in the right place or results may not be what you expect.

Notes

I used HiSoft's Devpac2 Assembler (also known as GenST Macro Assembler) to assemble the machine code for this auxiliary processor. HiSoft also supply a replacement file selector which is much better than the Atari offering. Further details from HiSoft at The Old School, Greenfield, Bedford MK45 5DE.

The assembler code must be relocatable, and it can be got into the workspace by reading the assembled file by the usual Native File-handling techniques, and dropping the first 28 bytes (the program header, which isn't required in APL). The auxiliary processor mechanism will accept the assembled code in any APL data format: I converted it to integer for use in <SETUP-FSELmc>, but for this program it is best sent to the AP in character, because you can manipulate characters more easily in the cover function.

If anybody wants to use these functions and doesn't want to run the gauntlet of keying all those numbers (and making all those mistakes!) I will be pleased to supply them with a suitable workspace if a diskette and the appropriate return postage is forwarded to me. For what it's worth, the assembler source code is also available.

Reference

[1] A description of the Auxiliary Processor Mechanism appears in a paper by Philip van Cleave in the APL83 Conference Proceedings. This paper is summarised at the end of Chapter 4 of MicroAPL's APL Language Manual.

APL Listings

Function FSEL

```

∇Z←PATH FSEL FNAME;AP;C;DIO
[1]  ♂ Call up the GEM AES File selector. Written 14 Mar 1989 by JS.
[2]  ♂ Allows the user to enter any garbage he/she likes.
[3]  ♂ If PATH or FNAME are null then use preferences from last time
[4]  ♂ (that's why the machine code for AP is kept in <FSELmc>)
[5]  ♂(2≠DNC'FSELmc')/'''VALUE ERROR: Variable <FSELmc> required''DERS 11'
[6]  ♂IO←0 ♂ →(0=ρFNAME)/Δ1
[7]  FSELmc[148+ι13]←13+(1 TRANSLATE FNAME),13ρDAV[0]
[8]  Δ1→(2≠DNC'PATH')/Δ2 ♂ →(0=ρPATH)/Δ2
[9]  FSELmc[84+ι63]←63+(1 TRANSLATE PATH),63ρDAV[0]
[10] Δ2←C←1 DSV0'AP DAT*' ♂ FSELmc[147 161]←DAV[0]
[11] AP←FSELmc ♂ C←AP ♂ Z←''
[12] →(C≤1)/0 ♂ →(~2|C)/0 ♂ Null vector if error or cancel
[13] ♂ If no filename then exit
[14] →(0=ρZ←(ZιDAV[0])←Z+0 TRANSLATE FSELmc[148+ι13])/0
[15] ♂ Try to decipher path
[16] C←(CιDAV[0])←C←(0 TRANSLATE FSELmc[84+ι63]),DAV[0]
[17] ♂(0≠+'\'=C)'/Z←((-⊖C)ι'\'')←C,Z ♂ →0'
[18] ♂ If we have no backslash we must have 'x:', so
[19] ♂(': '* (2+ C)[1])/Z←'' ♂ →0'
[20] Z←C[0 1],Z ♂ Drive-id, colon, filename ONLY
∇

```

Function SETUPΔFSELmc

```

∇SETUPΔFSELmc;A
[1]  ♂ This code initializes the machine code for the AP that calls the
[2]  ♂ GEM file-selector. Various strings are set to their defaults.
[3]  A←1610612896 1914195573 0 0 0 0 0 5898240 131074 0 0 0 0 0 0 0 0
[4]  A←A,1094343722 774504448 0 0 0 0 0 0 0 0 0 0 0 0 0 46268 16
[5]  A←A,1811939334 1912884853 1117332220 16 586942208 18408 533003
[6]  A←A,1172832288 650790376 2762442 1172832328 650790376 4728522 1172832332
[7]  A←A,650791400 5514444 1239941268 613172712 5514890 809238728 1312957096
[8]  A←A,4735617 1316290560 0
[9]  ♂
[10] FSELmc←4 DDR A ♂ Do it in character
∇

```

Assembler Code

HISoft GenST 680x0 Macro Assembler v2.07 14/06/89 17:51:24 Page 1
 FSELPRG.S - AES File Selector interface from APL.68000

```

3 T 00000000          * Written      13 Mar 1989 by JS
4 T 00000000          * Last amended 14 Jun 1989 by JS (coding tidied up!)
5 T 00000000
6 T 00000000          * Register usage
7 T 00000000          * Input
8 T 00000000          *   a0      Absolute address of <start>
9 T 00000000          *   a1      Pointer to result variable's header
10 T 00000000         *   a6      Base address of workspace
11 T 00000000         *   d2      Amount of free workspace
12 T 00000000         * Output
13 T 00000000         *   a6      Base address of workspace
14 T 00000000         *   d1      APL return code
15 T 00000000
16 T 00000000 600000A0 start bra ref Branch on reference
17 T 00000004 7218      moveq #24,d1 Nonce error for specification
18 T 00000006 4E75      rts No point hanging around here
19 T 00000006
20 T 00000020
21 T 00000020 005A0000000200020000 aespb ds.l 6 AES parameter block
22 T 00000048          control dc.w 90,0,2,2,0 FSEL function details
23 T 00000048          global ds.b 30
24 T 0000004C          intin ds.w 0
25 T 00000054          intout ds.w 2
26 T 00000054          addrin ds.l 2
27 T 00000054 413A5C2AZE2A00 addrout ds.l 0
28 T 00000094          path dc.b 'A:\*.+',0 Path to file-name
29 T 000000A1          ds.b 57 ... make this 64 bytes f.t.b.
30 T 000000A2          fname ds.b 13 File name
31 T 000000A2          even
32 T 000000A2          * Set up output variable first of all. This will be a scalar return code
33 T 000000A2
34 T 000000A2 B48C00000D10 ref cmp.l #16,d2 Have I got enough workspace?
35 T 000000A8 6C000006      bge wsok Yes - continue
36 T 000000AC 7204          moveq #4,d1 No - return Workspace Full
37 T 000000AE 4E75          rts No point hanging around here
38 T 000000B0
39 T 000000B0          * Prepare APL scalar for return code variable
40 T 000000B0
41 T 000000B2 4299          wsok clr.l {a1}+ Clear header of result
42 T 000000B2 22FC00000D10 move.l #16,{a1}+ Length
43 T 000000B8 22FC07000000 move.l #57000000,{a1}+ Integer scalar data type
44 T 000000BE
45 T 000000BE          * Now initialize the AES details
46 T 000000BE
47 T 000000BE 47E80008          lea aespb-start(a0),a3
48 T 000000C2 2208          move.l a3,d1
49 T 000000C4 45E80020          lea control-start(a0),a2
50 T 000000C8 26CA          move.l a2,(a3)+
51 T 000000CA 45E8002A          lea global-start(a0),a2
52 T 000000CE 26CA          move.l a2,(a3)+
53 T 000000D0 45E80048          lea intin-start(a0),a2
54 T 000000D4 26CA          move.l a2,(a3)+
55 T 000000D6 45E80048          lea intout-start(a0),a2
56 T 000000DA 26CA          move.l a2,(a3)+
57 T 000000DC 45E8004C          lea addrin-start(a0),a2
58 T 000000DE 26CA          move.l a2,(a3)+
59 T 000000E2 49E80054          lea path-start(a0),a4
60 T 000000E6 24CC          move.l a4,(a2)+
61 T 000000EB 49E80094          lea fname-start(a0),a4
62 T 000000EC 248C          move.l a4,(a2)
63 T 000000EE 45E80054          lea addrout-start(a0),a2
64 T 000000F2 26BA          move.l a2,(a3)
65 T 000000F4
66 T 000000F4          * Call the AES
67 T 000000F4
68 T 000000F4 303C00C8          move.w #20D,d0 AES magic number
69 T 000000F8 4E42          trap #2
70 T 000000FA
71 T 000000FA          * Save the return code (if any)
72 T 000000FA
73 T 000000FA 22A80048          move.l intout-start(a0),(a1)
74 T 000000FF
75 T 000000FE 42B1          clr.l d1 Clear APL error code
76 T 00000100 4E75          rts and go home.
77 T 00000102
78 T 00000102          end

```

A Vector Construction Kit

by Adrian Smith

Background

Over my spell as Editor of this excellent journal of ours, I have been gradually evolving a set of hardware and software tools to make my life as easy as possible. As I am now nearly to the end of this particular road, I thought I should document the current state of play, so that others may carry on where I leave off.

What you Need to Typeset APL

My inclination has always been to go for the simplest and most reliable tools for the job. Accordingly, you will find little in the way of high-performance, leading-edge stuff here! The basic requirements of an APL typesetting shop are in my view:

- you must see the APL characters on screen. In general, substantial APL bits (function listings) will be imported from workspaces, but it must also be possible to enter occasional characters (e.g. alt-251 for ρ) as required. Out of habit, I use the STSC $\square AV$, but APL2/PC would do rather better, as it stays away from the characters below blank (see below).
- text should wrap properly, even when you have bits of APL embedded in it. Again, it should be possible to see the final page layout on screen to a reasonable degree of accuracy.
- it should be easy to print the results, either in draft form on a cheap dot-matrix (e.g. an FX-80 compatible), or on a high-quality printer at at least 300 dots per inch.
- page numbering should be automatic, as should the creation of the contents list and Index to Advertisers.
- cut and paste should be kept to an absolute minimum. Ideally, all APL code should be set as part of the document, not clipped from submitted articles and stuck in.
- you do need to be able to create rectangular 'holes' for things like photographs, and have the text flow around them.
- APL code should look good on the page, and should be set big enough to ensure readability.

I suspect that I could have all sorts of fun with a Mac and a flash DTP package, but I doubt if the finished product would be half as effective! I would also spend far too long playing with the package!

Hardware Requirements


My current workstation consists of:

- an old IBM PC (the original twin-floppy kind) with a fast 33Mbyte hard disk and a 386SX upgrade card (including 4Mbyte of 16bit memory). Obviously the original power supply has been beefed up to 150 watts.
- a Hercules-Plus Graphics card, driving a monochrome VDU. This is ideal for text processing, and is much faster than either EGA or VGA running similar software.
- an Epson FX-80 (on LPT1:)
- an HP Deskjet+ (on LPT2:)

You can work nearly as well on an EGA or VGA system, as long as you have access to one of the common APL soft character sets, such as are shipped with APL*PLUS/PC or APL2/PC. However I definitely prefer the Hercules card, as it gives me true 'what you see is what you get' with bold, italic and so on, but at text-mode speed.

Software

I have a judicious mix of standard software and hand-crafted bits in APL and C:

- Microsoft Word 5. I have owned a copy of Word from version 1 upwards. I appreciate that it can be hard to learn, but I would say that it relates to other common word processors (such as Wordperfect) rather as APL relates to Cobol. In short, I like power and speed, and am willing to invest the time in learning how to use a powerful tool properly!
- GoScript Plus. This provides a full PostScript emulation for common or garden printers such as the FX80 and HP Deskjet. It will run on a plain old PC (given 640K of memory), but at an average speed of 10 minutes per page. With some EEMS memory and a maths co-processor (and a decent clock speed like 16MHz) it cracks along at a very respectable 50sec per page.
- APL2741.FNT. This is a complete APL PostScript font, modelled on the old 2741-golfball. It incorporates all the normal APL symbols, plus any oddballs like  that I could find. It is encoded to the APL*PLUS/PC layout, but with any characters below ' ' moved up above 128.

- APL*PLUS/PC. This is pretty well essential, as quite a lot of material arrives in .AWS format. For the same reason, I also have a copy of APL2/PC.
- APL2MSW.AWS. A bunch of functions to map the pathological characters (⊖ ⊙ ⊞ † ‡ ↔ and so on) and write out *EDIT* vectors as native text files with the necessary CR/LF pair at the end of each line.
- TF.EXE. A little C program, which will absorb almost anything (even a .AWS at a pinch) and output clean text. It has a couple of fancy bits to ensure that function listings don't get re-wrapped (and to deal with Wordstar files), otherwise it is a pretty basic filter. So far its only serious failure was when faced with a Multimate file, and it made a bit of a meal of Tony O'Hagan's article which was full of T_EX code.

Getting APL Characters on the Screen

In order to get APL characters on screen, there are two possible approaches, suitable for the EGA/VGA combination and the Herc+ system respectively.

1. For the colour systems, you can simply run any of the soft character sets (e.g. VGACHAR.COM from STSC or APL2FONT.COM from IBM) before you start Word. I actually use APL2FONT, patched to load the STSC character set, with the low-32 characters repeated above 128. The functions to do this were listed in Vector Vol.5 No.1 from page 85. The restriction is that you can only run Word in text mode, as the graphics characters are left untouched by the soft font. The page preview screen is also given a background of randomly jumbled characters (rather than a halftone grey); this is annoying, but appears to be harmless!
2. For the Hercules, you need to patch the font file that Word loads on invocation (called SCREEN.VID). This is in the normal Hercules format, so it is just a matter of using the same functions from Vector 5.1, but with an offset into the file. I tend to start Word with a batch file (e.g. WD V for Vector, WD C for correspondence, WD B for Gill's Brownie bits), so I only load the special font when I want Vector.

Snags

I have had very few problems with this approach. A couple of minor irritations are that Word uses the tiny centred dot (alt-250) as a space marker, and the normal centred dot (alt-249) in its ruler. APL thinks these are or and jot, so I either muck up my Word screen or have funny ors and jots.

Getting APL Characters on the Printer

This is beginning to sound like Computing at Chaos Manor (Jerry Pournelle's column in Byte). Again, I have two approaches to this:

1. For dot-matrix kit like the FX-80 and LQ-500, I use downloaded characters (thanks to Iain Hayward for the LQ set) and a little chunk of C to print the files. The two types of printer require slightly different strategies:

In the FX-80 you can load your character set into the characters above 128, however these then obliterate the built in italics! Accordingly, I load the APL characters into their normal $\square A V$ locations, and load *italic* characters into the locations of the normal upper and lower case. I then write out the print to file, load the character set, and run a little C program which simply passes the file straight through to the printer, except that whenever it encounters an Esc-4 (Epson code for italic on) it substitutes an Esc-%1 to switch into the download font. Similarly Esc-5 maps to Esc-%0 to reverse the effect.

In the LQ-500, you can only load characters into the lower 128. Accordingly, I have a slightly different C program which simply detects APL characters (using a boringly enormous *switch* statement) and prefaces these with a switch into the download font. It then writes out the appropriate character (e.g. α is mapped to A), and follows it up with the switch back into the normal character set. Overstrikes are written out as char-backspace-char, as there are obviously not enough holes in the bottom 128 characters for everything you need.

2. For decent quality print, I use GoScript, and simply write out a text file with Word set to its POSTSCRIP printer driver. GoScript has drivers for all the common printers (FX-80, LQ-500, HP Laserjet II etc), so all I need to be sure of in Word is that I have set the page length correctly. The APL font is exactly the same size as Courier (and is of course monospaced), so all I do in Word is set the APL bits to Courier-Oblique (so they look *italic* on the screen) at the appropriate size. Word is then quite oblivious of the APL, so all the line-wrapping continues to work as normal. The only patch required is to Word's PostScript prologue, where you must change Courier-Oblique to APL-2741; because this file is in plain editable text, you can do this with any basic text editor.

Snags

The first thing Word sends to the FX-type printer is an Esc-@ to reset it; this helpfully wipes out your download characters! I patched the printer driver to replace the @ with something harmless (5, I think). Word also attempts to make up the fancy characters like \acute{e} by e-backspace-acute; again I had to patch the printer table to stop this.

Obviously, you can't have your Courier in *italics* with the PostScript option. So far, I haven't felt the lack! A mildly annoying consequence is that as well as getting the *APL* text italic on screen, the symbols get sloped too ... so quad comes out as a parallelogram!!

Taste and Style

As I said earlier, I could have endless hours of fun with a top-line DTP system. However, when I look at the mess most of the common PC rags can make with Ventura, I think I shall stick to basics.

This is set in Palatino, set at 13 on 16 (before reduction). I have used very few tricks, other than the occasional drop-capital and some rectangular areas left blank for photographs. The big headings used as banners on the Education Vector are set at 48pt, with small caps; I suspect that this is about as big as you can go without getting too silly! The APL is set at 14pt in the text, and 12 on 13 in the listings. I think this is about right.

If you want lots of silly effects (like APL listings written vertically up the page) I would be delighted to produce them. Sensible suggestions on the format will obviously be welcomed, and may even have some effect. For the moment, I will concentrate on the content, and hope that the presentation is just good enough that nobody really notices it!

A Note on Quad and Quote-Quad

by Joseph L.F. De Kerf

At the APL-ication Conference, M.T. Wheatley [1] presented a paper on "Extending the Domain of APL". In a footnote he states: "Formally, the names of system functions and variables may begin with either \square or \square , however, to date no \square names, other than \square itself, have been introduced."

This is true, as far as IBM implementations of APL are concerned, but distinguished names beginning with the Quote-Quad \square have really been implemented, namely in 4000 APL (General Electric Computers GEC) [2] and in WATCOM APL (WATCOM Products) [3].

4000 APL supports two modes of file handling: component files, which are intended only for access by APL, and logical files, which enable the user to access files created or processed by other languages. Distinguished names related to the use of component files begin with the Quad \square , while distinguished names related to logical files begin with the Quote-Quad \square . The following logical file system functions are available:

\square <i>CNTRL</i>	Executes a data management control command
\square <i>CONN</i>	Connects a file and associates a tie number
\square <i>CREATE</i>	Creates a file or catalogue
\square <i>DCONN</i>	Connects a device and associates a tie number
\square <i>READ</i>	Reads a record from the specified file
\square <i>REPL</i>	Replaces a record in the specified file
\square <i>SEEK</i>	Seeks for a record or key in the specified file
\square <i>WRITE</i>	Outputs a record to the specified file

WATCOM APL too supports component and native files. Distinguished names related to both modes of file handling begin with the Quad \square , except the names of the system functions to perform character stream input and output for native files operations, which begin with the Quote-Quad \square :

\square <i>GET</i>	Get character string from the specified file
\square <i>PUT</i>	Put character string to the specified file

It should be noticed that in some implementations of APL such as APL*PLUS/PC et al (STSC) [4], the distinction between the distinguished names related to component and native files is detectable by the fact that the second character in the name is the letter *F* or *N* respectively, e.g. \square *FREAD* and \square *FWRITE* for component files versus \square *NREAD* and \square *NWRITE* for native files. This may be less remarkable, but it avoids the use of an overstruck character.

References

- [1] M.T. Wheatley: *Extending the Domain of APL*; APL-ication Conference Proceedings, University of Kent, England, 28-30 September 1988; British APL Association BAA, London, England, 1988, pp.1-7.
- [2] *APL For OS4000 User Software Handbook*: Publ. No. DD 1558; GEC Computers Limited, Borehamwood, Hertfordshire, England, October 1980.
- [3] J.C. Wilson and T.A. Wilkinson: *WATCOM APL Users' Guide (IBM PC with DOS)*; WATCOM Publications Limited, Waterloo, Ontario, Canada, May 1983.
- [4] *APL*PLUS System for the PC - Reference Manual*: STSC Incorporated, Technical Documentation, Rockville, Maryland, 1986.

British APL Association Software Library Software Submission Form - Page 1 of 2

Please copy and fill in this form for EACH disk you submit.

Details corresponding to items flagged (*) will NOT be made publically available, but are for our records only. Please Use BLOCK CAPITALS for all items except number 18.

0. Submission date: _____
1. Name: _____ 2. Daytime phone(*): _____
3. FULL address(*): _____

4. Electronic mail addresses(*): _____
5. Short disk title: _____
6. Brief description of disk: _____

7. List target machine: _____
8. List additional software required: _____
9. Indicate special hardware requirements: _____
10. Is user documentation provided on the disk?(Y/N): _____
11. List titles of any paper documentation included with your submission:

12. Is this documentation, or any other, available to users upon application to you? (Y/N): _____ Please give details: _____

13. Does the disk include any form of payment request from users of the software?(Y/N): _____ Please give details: _____

14. Does your submission constitute a 'demonstration disk' in that it demonstrates software that is available for purchase?(Y/N): _____

British APL Association Software Library Software Submission Form - Page 2 of 2

- 15. If the TARGET machine is NOT a DOS-based PC, does the disk include instructions for transfer to the target machine?(Y/N):_____

- 16. File names and descriptions. This information will be made publically available in the software library catalogue. Please save us some work by including these details on a file named <CAT> (no file extension) on your submitted disk. Please enter these details for all files on the disk, except <CAT> itself. You can affix a print of <CAT> below.

For APL workspaces, please use the space below to document functions.

FILENAME.EXT	SHORT DESCRIPTION

- 17. Use any extra space above for comments you wish to appear in the software library catalogue.

- 18. Your signature is necessary. It declares your legal right to make the disk freely available for copying and use, and grants that right to the British APL Association.

Signature: _____

Mail your submission to: The BAA Software Library, c/o Alison Chatterton, 9 Oak Grove, Hertford, Herts SG13 8AT, ENGLAND

BAA Software Library: Order Form

To: Alison Chatterton
 9 Oak Grove, Hertford,
 Herts SG13 8AT, ENGLAND

Please supply the disks circled below:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99	100							

Total number of disks (please check): _____

at 2 Pounds per disk: BAA Member's rate _____
 OR at 3 Pounds per disk: Non-member's rate _____

I want to join the BAA, and enclose
 the 10 Pounds annual membership fee: _____

Postage and handling: 1.00

Add 2 Pounds for orders outside the UK: _____

Total order - remittance included: _____

Your signature: _____

Your name: _____

Full address: _____

Please make cheques payable to the British APL Association. Payment can be made in \$ - pay 1.5\$ (US) for each £. All orders must include payment - allow 30 days for delivery.

Software is accepted by the British APL Association on good faith and we do not vouch for or make any claims regarding donated software. The BAA shall not be held responsible for damage caused by the use of library software, and shall not be liable in any way in the event that submitted material proves to be the subject of third party copyright. Your signature above accordingly indemnifies the BAA from all liability.

Index to Advertisers

APL People	2
British Airways	68,69
Cocking and Drury	6
Dyadic Systems Ltd	8
HMW Computing	48
Mercia Software (half)	5
MicroAPL (2xhalf)	88
APL Booklist (Renaissance Data Systems)	36

All queries regarding advertising in VECTOR should be made to Alison Chatterton, at the address on the inside back cover.

BAA: Membership Application Form

Membership of the British APL Association is open to anyone interested in APL. The membership year runs from 1st May to 30th April.

Name: _____
 Department: _____
 Organisation: _____
 Address Line 1: _____
 Address Line 2: _____
 Address Line 3: _____
 Post or zip code: _____
 Country: _____
 Telephone Number: _____

Membership category (please circle): 89/90

Non-voting student membership (UK only) £5
 UK private membership £10
 Overseas private membership £18
 Airmail supplement (not needed for Europe) £8
 Corporate membership £85
 Corporate membership overseas £140
 Sustaining membership £360

For student applicants:

Name of course: _____
 Name and title of supervisor: _____
 Signature of supervisor: _____

If you would like to pay by credit card ...

Card no (Access/Visa): _____

PAYMENT

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

BAA Administration,
 Alison Chatterton, 9 Oak Grove,
 HERTFORD SG13 8AT
 England

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.

1989/90 Committee

Chairman:	Peter Donnelly 0420-87024	Dyadic Systems Ltd., Park House, The High Street, ALTON, Hants GU34 1EN
Secretary:	Graham Parkhouse 0483-571281 (x2379)	Dept. of Mech. Eng., University of Surrey, GUILDFORD GU2 5XH.
Treasurer:	John Sullivan 01 462-3628	24 Mounthurst Road, Hayes, Kent BR2 7QN
Journal Editor:	Adrian Smith 04393-385	Brook House, Gilling East, YORK YO6 4JJ
Activities:	Dr Peter Branson 0737-242950	Oaklands Cottage, Wray Common, Reigate, Surrey RH2 0LE
Education:	Dr Alan Sykes 0792-295296	Dept of Management Science University College of Swansea, Singleton Park, SWANSEA SA2 8PP
Publicity:	Jonathan Martin 01 562-5697	(S499) British Airways, PO Box 10, Heathrow Airport, HOUNSLOW, Middlesex TW6 2JA
Technical:	David Eastwood 01-922 8866	MicroAPL Ltd South Bank Technopark, 90 London Road, LONDON SE1 6LN
Recruitment:	Jill Moss 0225-462602	APL People Ltd, The Old Malthouse Clarence St., BATH, Avon BA1 5NS
Projects:	John Searle 01 948-6737	13A Mount Ararat Road, Richmond, Surrey TW10 6PQ
Administration:	Alison Chatterton 0992-552489	9, Oak Grove, HERTFORD, SG13 8AT

Journal Working Group

Jonathan Barman	048839-575
Anthony Camacho	0727-60130
Cathy Dargue	0895-31313
Adrian Smith	04393-385
David Ziemann	01-436 9481

Typeset using MS Word 5.0 and GoScript on HP Deskjet+

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 many timesharing bureaux and on most mainframe, mini and micro-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.

APL People
The Old Mailhouse
Clarence St. BAYFOLK, Essex SBN
Tel: 0225-462602

Cocking & Drey Ltd
459 Tottenham Court Rd
LONDON, W1P 9LE
Tel: 01-493 0484

HMW Computing Ltd
Hamilton House,
4 Temple Avenue,
LONDON EC4Y 0HA
Tel: 01-493 4212
Fax: 01-493 3625

Imper Product Ltd
Eagle House
78 Clapham Common Southside
LONDON SW4 9DG
Tel: 01-873-3854

MicroAPL Ltd
South Bank Technopark
90 London Road
LONDON SE1 7 5LN
Tel: 01-822 4866

Peter Cymax Systems
213 Goldhurst Terrace
LONDON NW6 3ER
Tel: 01-824-7013
Mobile: 0181 877966

Intelligent Programs Ltd
9 Gun Wharf, 130 Wapping High St
LONDON, E1 9NH
Tel: 01-265 1120

Dyadic Systems Ltd
Park House, The High Street
ALTON, Hants
Tel: 0420-67024

APL Impetus Ltd
Rusper, Sandy Lane
by Hatch, SEVENOAKS
Kent TN15 0FD
Tel: 0732-685126

Mechanics Systems Ltd
656 West Barnes Lane
Molesey Park, Surrey
Tel: 01-839-6298
Fax: 01-836-1062

IP Sharp Associates
1-4 Singer St
LONDON EC2A 4BQ
Tel: 01-837 1186
Fax: 01-837 9722

