Programming for Scientists (...and engineers!-)

Alex Martelli (aleax@google.com)

http://www.aleax.it/scipy_key08.pdf

I'm no scientist, myself...

Google

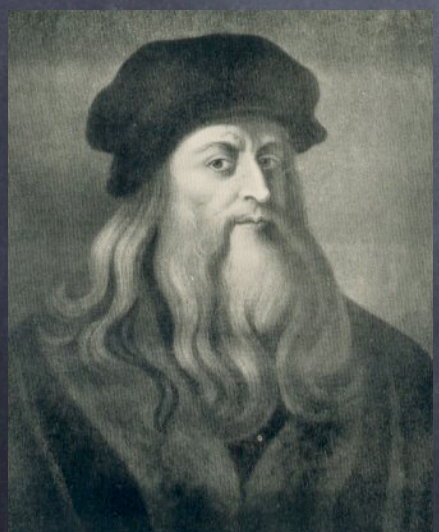
I'm no scientist, myself...

- ...nor do I play one on the net!
  - if anything, I play the philosopher, the historian, the manager, the linguist, the businessman, the bon vivant...;-)
- but, what I AM, is -- an engineer!
  - originally EE, just "drifted" into SW...;-)

Google

...I'm an engineer!

- so, to interpret my remarks, remember...:
- engineers are practical and results-driven
  - just like Wittgenstein, Korzybski, Saint-Exupéry (well he TRIED!-), Da Vinci, Scott Adams, Rowan Atkinson, Frank Capra...

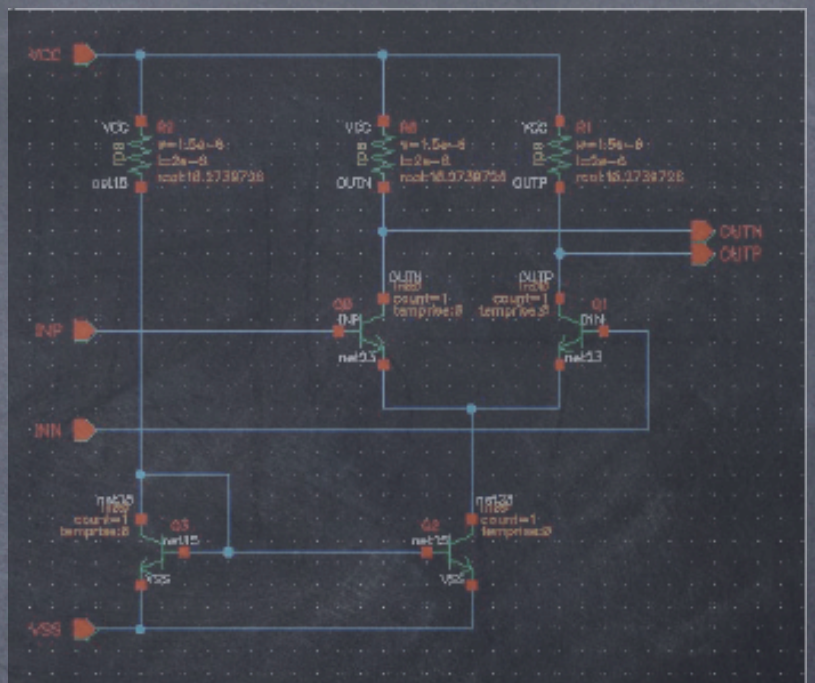


Google

So, why did I program...?

- circuit simulation? but, SPICE ruled!-)

```
VCC 7 0 12
VEE 8 0 -12
VIN 1 0 AC 1
RS1 1 2 1K
RS2 6 0 1K
Q1 3 2 4 MOD1
Q2 5 6 4 MOD1
RC1 7 3 10K
RC2 7 5 10K
RE 4 8 10K
.MODEL MOD1 NPN BF=50 VAF=50 IS=1.E-12
RB=100 CJC=.5PF TF=.6NS
.TF V(5) VIN
.AC DEC 10 1 100MEG
.END
```



- (it STILL rules, btw;-)

Some low-level stuff...

- "some assembly required", of course
  - to understand the microprocessor
  - and design systems/circuits around it
- and, of course, some microcode
  - the line between "FPGA design" & µcode is sometimes quite blurry!-)

Google

But, the real reason...

- "personal interest in combinatorics"
  - that's resume-speak for...:
  - probability & my hobby: contract bridge
  - so, mostly $\binom{n}{k}$ binomial coefficients;-)
- unfunded computing in the '70s was quite a problem...;-)
- 1977: a young professor takes pity on a bunch of undergrads, lends them punched cards w/secret codes so they can run Fortran jobs (not just compile them)...

Google

Programming in 1977

```
      FUNCTION KOFN(K, N)
      INTEGER*4 I, K, N, K1, N1
      INTEGER*8 KOFN, NRES
      K1 = K
      N1 = N-K
      IF(K1.GT.N1)THEN
        K1 = N-K
        N1 = k
      ENDIF
      NRES = 1
      DO 10 I = N1+1, N
        NRES = NRES * I
10     CONTINUE
      DO 20 I = 2, K1
        NRES = NRES / I
20     CONTINUE
      KOFN = NRES
      RETURN
      END
```

Programming in 2008

```
f2py -c -m kofn kofn.f
```

or (maybe w/import psyco; psyco.full()...):

```
def kofn(k, n):
    nres = 1; k1 = k; n1 = n - k
    if k1>n1: k1, n1 = n1, k1
    for i in range(n1+1, n+1): nres *= i
    for i in range(2, k1+1): nres /= i
    return nres
```


Google

What about runtimes...?

```
$ python -mtimeit -s'from kofn import
kofn' 'kofn(4, 13)'
1000000 loops, best of 3: 0.627 usec per

$ python -mtimeit -s'from pkofn import
kofn' 'kofn(4, 13)'
100000 loops, best of 3: 4.24 usec per

$ python -mtimeit -s'from pkofn_psy import
kofn' 'kofn(4, 13)'
10000000 loops, best of 3: 0.106 usec per
```

Google

Or, for #s beyond 64bit:

```
$ python -mtimeit -s'from pkofn import kofn'
'kofn(13, 52)'
100000 loops, best of 3: 14.6 usec per loop
$ python -mtimeit -s'from pkofn_psy import kofn'
'kofn(13, 52)'
100000 loops, best of 3: 7.01 usec per loop
$ python -mtimeit -s'from pkofn_gm import kofn'
'kofn(13, 52)'
100000 loops, best of 3: 2.76 usec per loop
```

but, NOTE...:

```
$ python -c'from kofn import kofn; print kofn(13, 52)'
1066226105
$ python -c'from pkofn_gm import kofn; print kofn(13,52)'
635013559600
```

btw, ..._gm's simple...:

```
import gmpy

def kofn(k, n):
  return int(gmpy.comb(n, k))

# http://code.google.com/p/gmpy/
```


Google

So, black boxes rule...?

- "Dammit Jim, I'm a...
  - doctor
  - chemist
  - physicist
  - computer scientist
  - civil engineer
  - sociologist
  - economist
- ..., not a mathematician!"
- means: "I don't WANT to HAVE TO CARE...!"

Google

as I was first teaching it...

- http://www.ima.umn.edu/~arnold/455.f96/disasters.html
- inaccurate calculation of the time since boot due to computer arithmetic errors
- 24-bit fixed-point register for 1/10.0, but:

```
$ python -c'print repr(1/10.0)'
0.10000000000000001
```

- no finite binary repr for 1/10.0...
- in 100 hours, accumulates error enough to let the incoming target travel 0.5 Km...

Google

Feb 25, 1991

Google

So, DO black boxes rule...?

- short answer:

NO!

Google

So, DO black boxes rule...?

- longer answer:

NO WAY!

Google

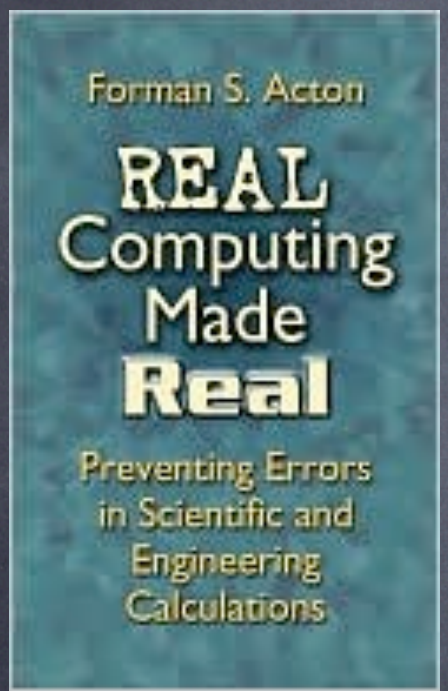
So, DO black boxes rule...?

- even longer, well-reasoned answer...:

nature and numbers are not in the business of making your life simpler by ensuring your computations are always well conditioned and well behaved; dumping all your numbers into a black box won't make them so -- in the end there's just NO substitute for actually understanding what you're doing; ANY numeric package will NEVER perform said understanding *on your behalf*

Google

And, Forman Acton agrees...



- $12.76 at Amazon (!)
- 200 pages + exercises' solutions
- zero code, or, close (p. 155-6);-)
- "visualize the shape of the function, and use that shape to construct your algorithms"
- "the purpose of computing is insight, not numbers!" (Hamming)

Google

Simple example fm Acton

- $\sqrt{(x+1)} - \sqrt{(x)}$ where x is large...
- so just multiply and divide by $\sqrt{(x+1)} + \sqrt{(x)}$
- (A - B) * (A + B) / (A + B) -> $(A^2-B^2)/(A+B)$
- $\frac{1}{\sqrt{(x+1)} + \sqrt{(x)}}$ is much better for any x>0!

Google

Avoid 'blahs'!

```
from math import sqrt
def fblah(x):
  return sqrt(x+1) - sqrt(x)
def fyeah(x):
  return 1.0/(sqrt(x+1) + sqrt(x))
x = 9876543.21
print fblah(x), fyeah(x)
# 0.000159099021857 0.000159099021739
# already gaining 3 significant digits!
```


Google

Acton's pearls of wisdom

- software has bugs -- often quite obscure bugs that bite only rarely
  - this kind of software is called "reliable" or "mature"

Google

Acton's pearls of wisdom

- if the numbers you are producing are important, compute them twice - by different algorithms
- (IMHO: best advice ever for ANY batch-mode computation, numerical or otherwise, if you can possibly afford it...)

Google

Acton's pearls of wisdom

- the longer I have computed, the less I seem to use Numerical Software Packages (except for linear algebra packages) -- packages inevitably hide deficiencies in a problem's formulation (poorly conditioned equations, unexpected singularity, ...)
- (i.e., *no black boxes* -- even linear algebra packages need some tiny amount of wisdom and understanding by their user!-)
  [e.g. $\frac{1}{3}N^3$ multiplies & adds for Gaussian elimination, vs $3/2\ N^3$ for solving via inverse found by Gauss-Jordan...]

Google

If I compute OK all is rosy

- ...you wish (in most disciplines)...!-)
- in many disciplines, sampling bias kills you
  - meta-issue: publication bias (the "file drawer problem") can give meta-level bias to whole fields in ANY discipline!-)
- how good are your numbers in the first place? (measurement, design of experiments)
- are you controlling for all the right factors (e.g.: correlation of time series)...?

Google

L'envoi

- Spolsky's Law of Leaky Abstractions:
  - "All non-trivial abstractions, to some degree, are leaky".
- so that's the problem with black boxes: they're abstractions -- useful, BUT...
  - they're leaky
  - you need to know what's inside
  - you need to CARE
  - you need to learn what to DO about it
- numerics, sampling, measurements, ...!

Google

Key points to retain...:

- engineers are not the same as scientists, and might not fit your preconceptions well
- today, you can code in ways that are fast, elegant, clear AND numerically efficient
  - but, to get there, you have to CARE...
- black boxes are a necessary evil
  - we need them for reasonable productivity
  - but, being abstractions, they DO leak, so,
  - you still have to know what you're doing

Google

http://www.aleax.it/scipy_key08.pdf

Q?

A!

Google