BUGS, ETC

Note: this talk is off the record. I may find it necessary to disavow any remarks that may be attributed to me.
ESTIMATED FREQUENCY
OF BUGS

1 PER 50 LINES
(early phase)

1 PER 1000 LINES
(after intensive effort)

THEOREM

If the code has more than 5000 lines
it gives the wrong answer
with probability one
PRESSURES FOR COMPLEXITY
COMPLEXITY IN AIRCRAFT

Cost projection:

\[ \text{GDP} \]

\[ \text{Cost of 1 military airplane} \]

\[ \rightarrow \text{Year} \]

2050

GDP will be sufficient for 1 airplane in 2050

Pressures for complexity:

1). DOD has departments to generate requirements. It can be estimated that the number of requirements will be roughly proportional to the number of people employed to think of them.

2). 2000 sub-contractors each have an interest in introducing more complex subsystems.

3). A design staff of 1000 have to occupy themselves somehow (35 designed the Mirage 3)
COMPLEXITY IN COMPUTER PROGRAMS

Is there a danger that a computer program will require the GNP, say, by the year 2100?

Pressure for complexity:

1) One might imagine that the purpose of computers is to replace human effort by machine effort. The goal of numerical analysts is precisely the opposite!

2) The objective of professors and research scientists is generally to show how clever they are. This objective is not realized by a simple solution — that merely suggests that the problem was easy.

3) A solution terminates funding — the secret of funding is the light at the end of the tunnel.
CASE HISTORIES

OF BUGS
BUGS

CRANKED WING  FLO22  1976-1983

SWEPT FORWARD WING  FLO27  1977  6 weeks

EXTENDED WING TIP  FLO57  1981-1982

MISSING TAILPLANE  FLO59  1983

(MULTIGRID)

WRONG CALLING SEQUENCE  FLO52  April 1981

MISSING DO STATEMENT  FLO57  1983  3 months
CRANKED WING BUG

FLO22 1976-1983

Missing scale factor for
\[ \tau_0(x), \gamma_0(x) \]

Found by J. Chung Chung / D. Coupland
THE CRANKED WING FAILURE

1) I tried a change of scale (GOOD), but I did it with a straight leading edge (BAD – not exercising all the options)

2) I expected FAILURE
   - I assumed a KINK would invalidate the assumptions of the coordinate transformation

3) I attributed bad results for TACT wing to
   (1) Singularity in the transformation
   (2)Sparse mesh on outer wing

   These were WRONG PLAGUSIBLE EXPLANATIONS
SWEPT FORWARD WING

FLO 27 1977

Calculated wing

Intended wing

Root

Pressure distribution
EXTENDED WING TIP

FL957 1981-1982

On the wing

IF (K < L T. K TIP)

not IF (K L E. K TIP)
MISSING TAILPLANE

FLO 59 1983

Flares recalculated for tailplane cells
Not transmitted to this subroutine
INSIDE OUT PROBLEM

FLO57, FLO59

What's sign of projected force area (surface normal)

Wrong sign on all forces

April 1981 24 hours

Inside out tail

1983 6 months
**THE SIGN PROBLEM**

**KORN'S LEMMA**

The sign is either **POSITIVE**,

or else it is **NEGATIVE**.

When in doubt, **TRY BOTH**.
DEBUGS (1)
(Frequently not performed by me)
when they should have been

CONSISTENCY CHECKS

Double the scale of the profile
Insert uniform flow
Check symmetry with symmetric flow
Check convergence with decreasing mesh size

PROGRAMMING CHECKS

Exercise all options
Set core to indefinite
EXTERNAL CHECKS

Check against known exact solutions
Check plausibility of result
Check against experimental data
Plot everything

ACCIDENTAL DISCOVERY

When coding new versions

By third parties
- release pilot code
to carefully selected friendly users

By concentrated thinking
- it pays to carry the whole code in your head
DEFENSIVE PROGRAMMING

1) MODULAR STRUCTURE
   - change one thing at a time

2) DUAL PATH PROGRAMMING
   - interchangeable subroutines in each segment

3) SIMPLICITY: avoid
   (1) AICS
      (acquired if contamination syndrome)
   (2) RSNS
      (random statement number syndrome)

4) RIGID PROTOCOL
MODULAR PROGRAMMING

GRID GENERATING MODULE

INPUT MODULE

FLOxx

OUTPUT MODULE

EULER

BCWALL

BCFAR

FILTER
DUAL PATH PROGRAMMING
EFFICIENCY SIMPLICITY TRADE-OFF

Example: SYMMETRIC MATRICES

Complete Storage

Reduced Storage (1)

Reduced Storage (2)

One certain result: schemes (1) and (2) lead to INCOMPREHENSIBLE CODE