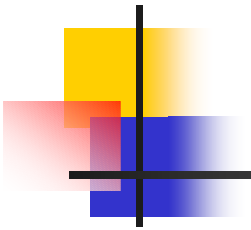
“goto considered not particularly harmful: defect analysis in a major numerical library"

Les Hatton

CISM, Kingston University
L.Hatton@kingston.ac.uk

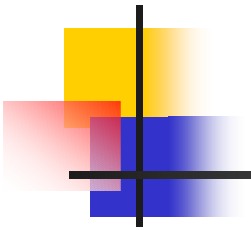
Version 1.1: 04/Jun/2008

Overview



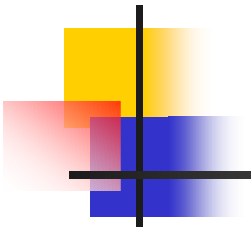
- Some background
  - A little bit about defects
  - Software metrics and a bit of history
  - PCA (Principle Component Analysis)
- The NAG Fortran library
- Defect analysis
- Conclusions

A little bit about defects



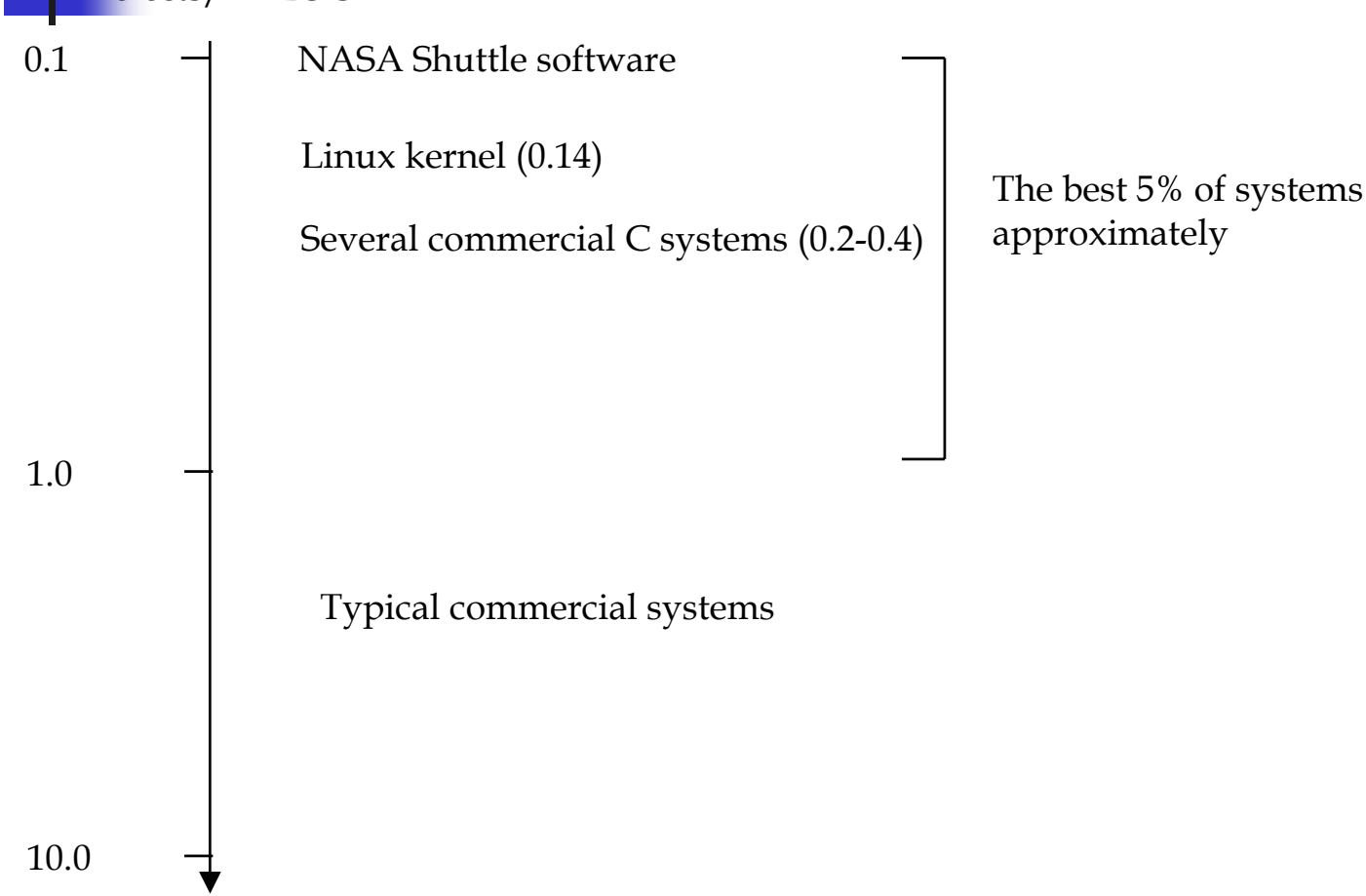
- A *fault* is a statically detectable inconsistency in code or a design
- A *failure* is any difference at run-time between expected and actual behaviour
- A *defect* is a fault that has failed
- Every failure is caused by at least one fault
- Not all faults fail

A little bit more about defects

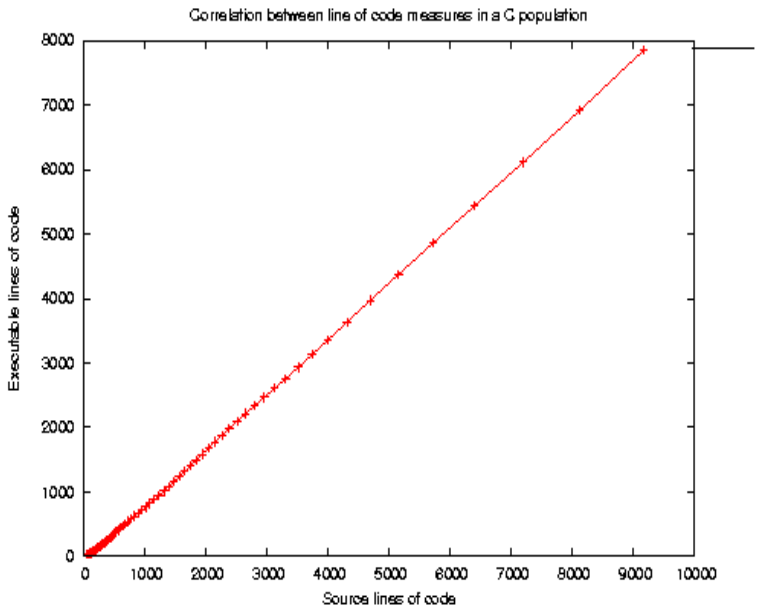


- If your software exhibits less than one defect per KXLOC (thousand executable lines of code) in its entire life-time, it is about as good as anything ever has been.

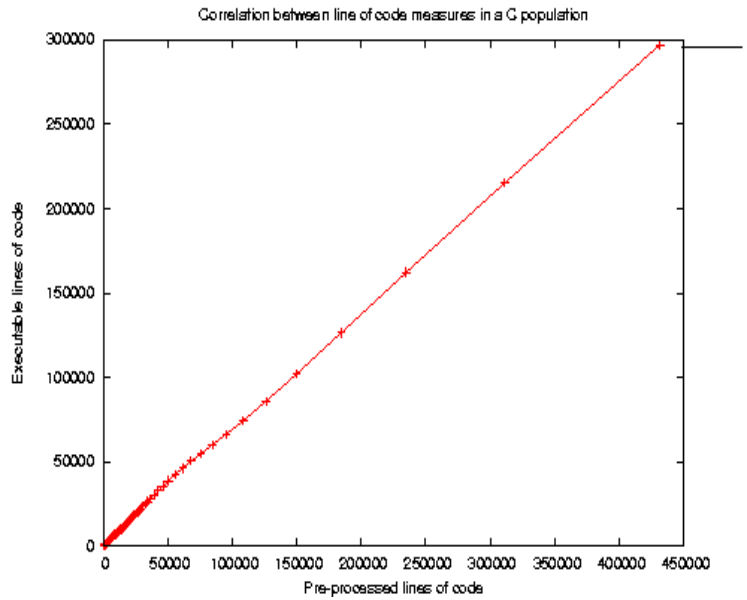
A software quality scale based on defect density



Alternative line measures in C

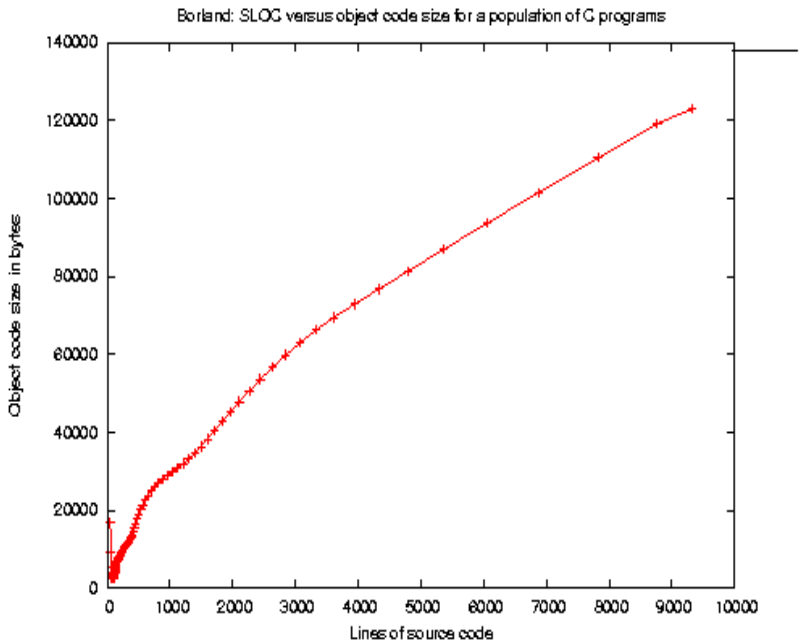


SLOC v. XLOC

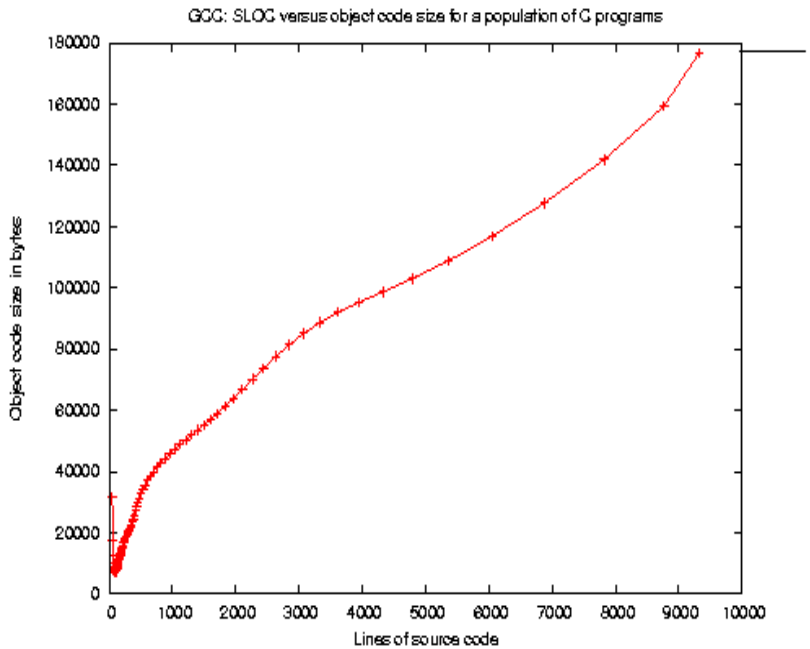


PPLOC v. XLOC

Relationship between SLOC and object code size in bytes



SLOC v. Object bytes (Borland)



SLOC v. Object bytes (gcc)

Estimating LOC from object code size

From/ To	SLOC	PPLOC	XLOC	Bytes
SLOC	1.0	1.24 +/- 0.11	0.87 +/- 0.03	17 +/- 3.5
PPLOC	0.81 +/- 0.07	1.0	0.7 +/- 0.04	14 +/- 4.0
XLOC	1.15 +/- 0.04	1.42 +/- 0.08	1.0	19.5 +/- 4.2
Bytes	0.058 +/- 0.01	0.071 +/- 0.16	0.051 +/- 0.08	1.0

Example: estimating Windows SLOC

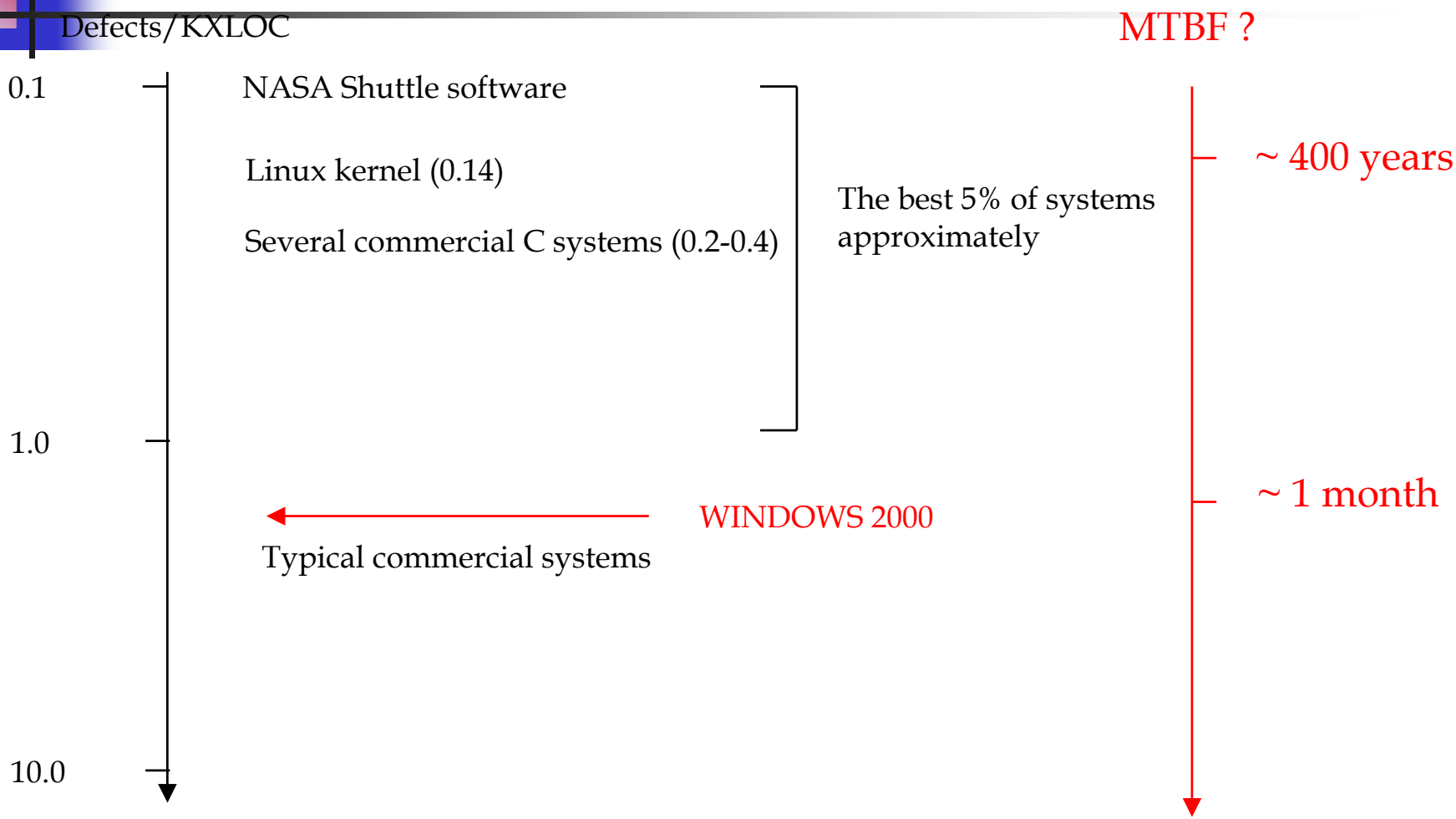
OS	Size in object code (Mb.)	Estimated SLOC (MSLOC)
Windows 98	**174**	**9.9 +/- 1.6**
Windows 2000	**539**	**30.8 +/- 2.9**
Windows XP	**699**	**39.9 +/- 4.9**

Example: estimating Windows defect density

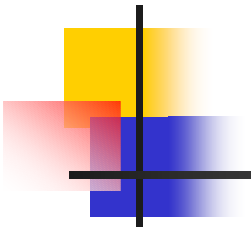
OS	Quoted defects	Estimated XLOC (MXLOC) (conversion factor 0.87)	Estimated defect density
Windows 2000	**63,000***	**26.8 +/- 2.5**	**2.35 / KXLOC**

* http://www.archives.cnn.com/2000/TECH/computing/02/17/win2k.bugs.idg

A software quality scale based on defect density



Software metrics and a bit of history

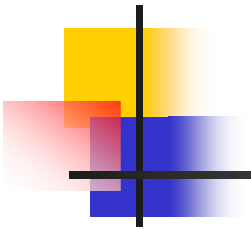


- Quoted examples in the literature
  - Cyclomatic complexity (decision count)
  - Myer's interval (complex decision count)
  - goto
  - if .. else if .. but no else statement
  - Halstead metrics (loosely based on Shannon information theory)
  - Knots (crossings of control-flow, i.e. non nested)
  - Number of unused variables
  - Maximum nesting of decisions
  - Granny's birthday …

Principle Component Analysis

- Multi-dimensional regression
- Defines the “shape” of the n-dimensional data cloud.
  - Long and thin means a linear combination of the independent variables is highly correlated with the dependent variable
  - Chubby and relatively isotropic indicates no special combination is dominant
- Automatically rotates the axes to align along the “principle” direction.

Software metrics and a bit of history



- It is widely believed that these correlate with defects.

Principle Component Analysis

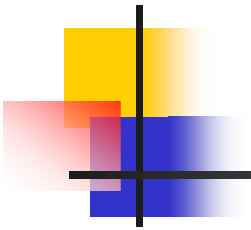
$Y = PX$

(X is mxn matrix of original data, m number of independent variables with n observations)

We are looking for P such that YY^T is diagonalised, i.e. a rotation and stretch so that the covariance matrix of Y is in its simplest form

This turns out to be $P = E^T$ (E is the matrix of eigenvectors of XX^T)

Overview

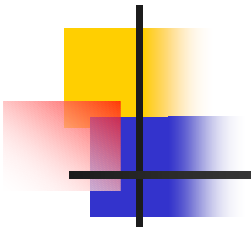


- Some background
- The NAG Fortran library
- Defect analysis
- Conclusions

The NAG Fortran library

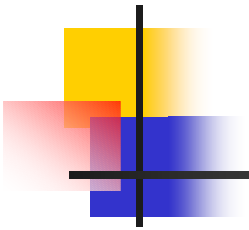
- 19 versions (marks) 1978-1999.
- 266,123 XLOC; 3,659 subroutines
- Defect history is extractable using Perl scripts from its headers.
- Defect density 4.9 / KXLOC

The NAG Fortran library - analysis



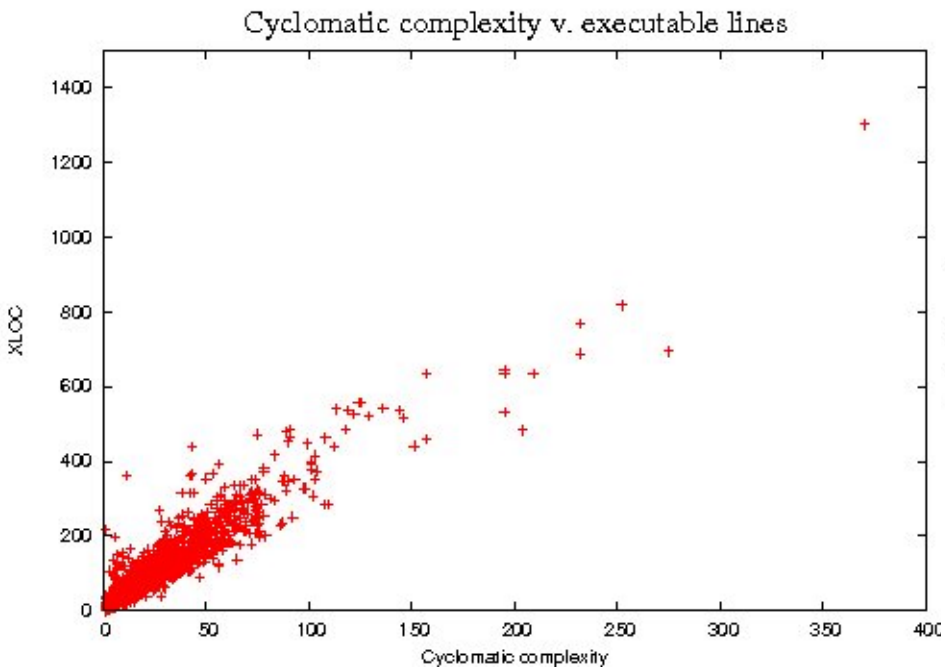
- Complete ISO Fortran 77 parsing engine built
- 15 “metrics” extracted from the 3,659 components.

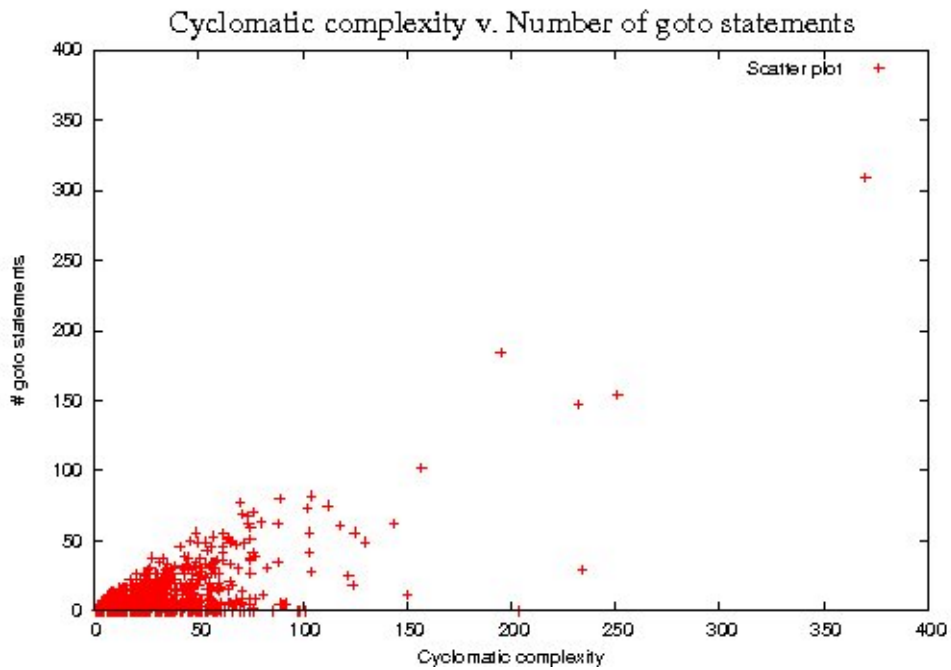
Overview



- Some background
- The NAG Fortran library
- Defect analysis
  - Metric correlations
  - PCA results
  - More evidence of scale-free behaviour
- Conclusions

Metric correlations

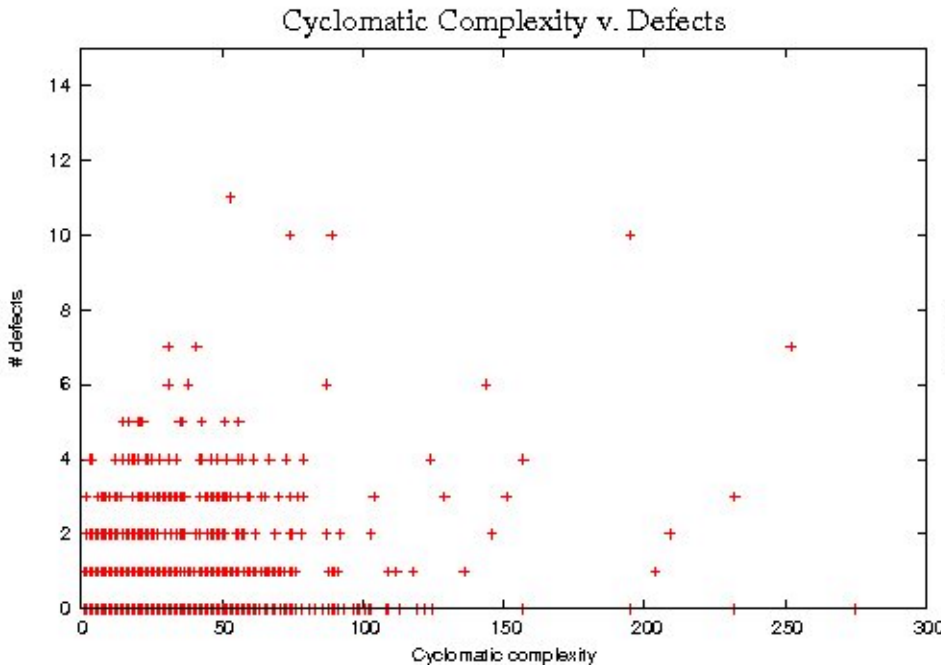




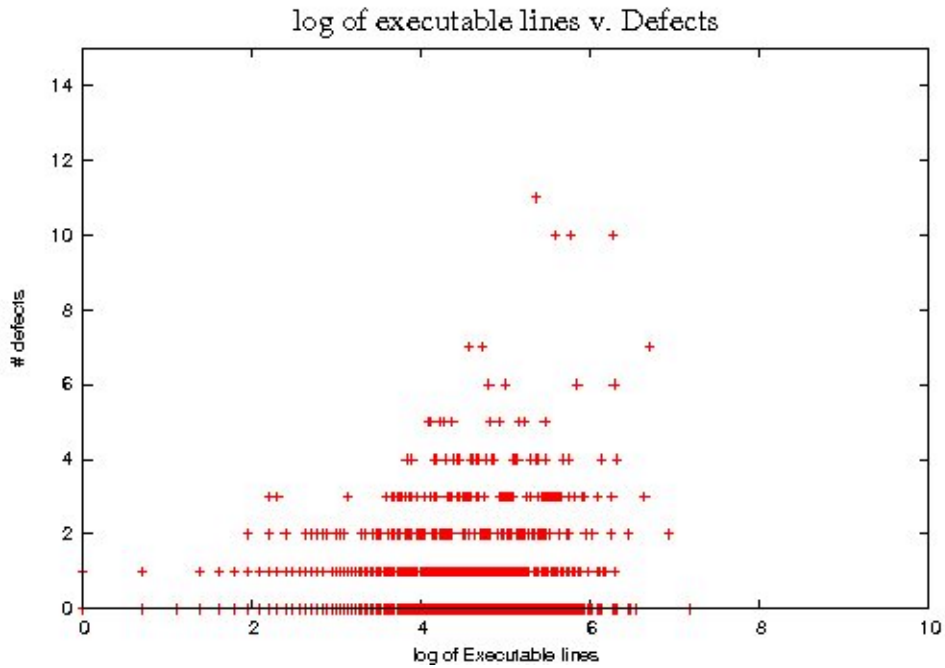
Cyclomatic complexity v. XLOC

Cyclomatic complexity v. goto

Defect correlations

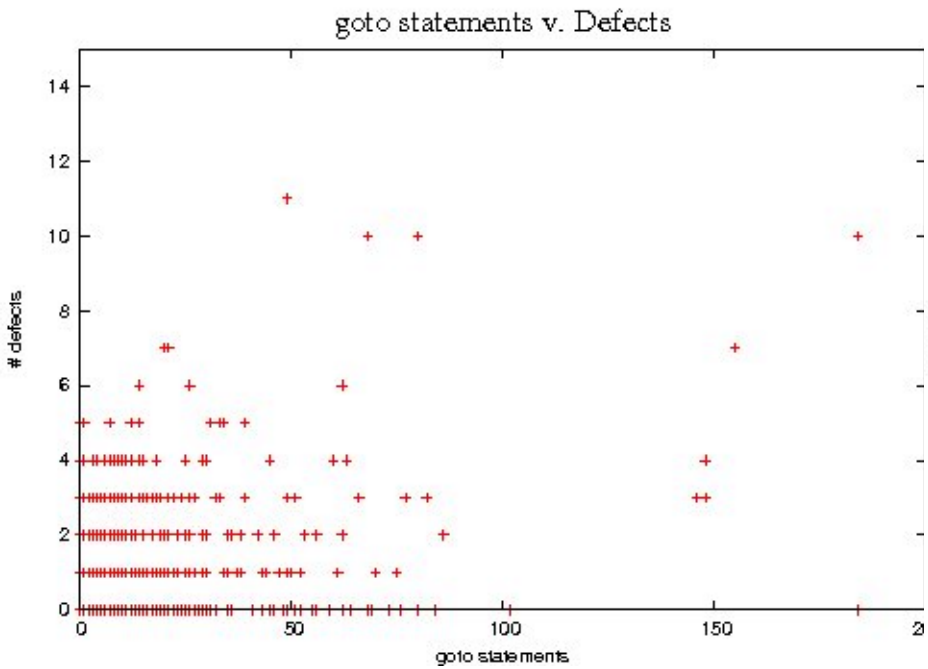


Cyclo. complexity v. defects

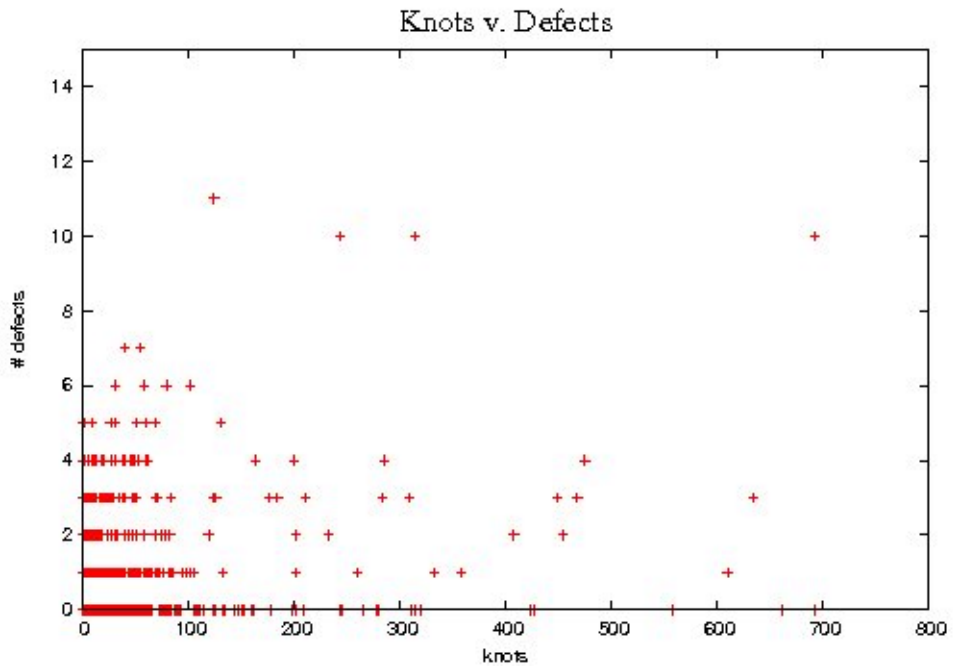


Ln(xloc). v. defects

Defect correlations

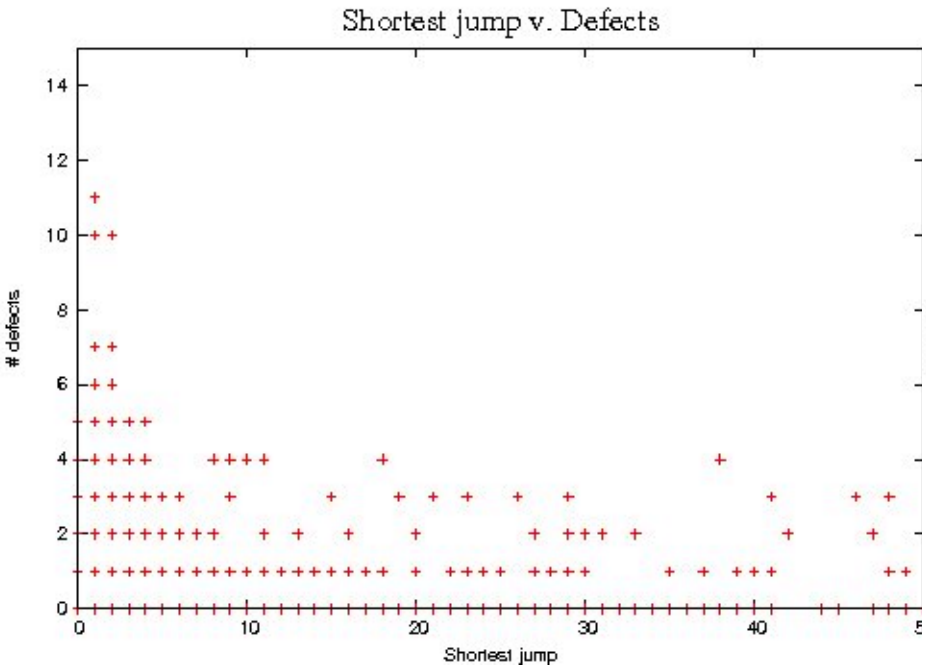


goto v. defects

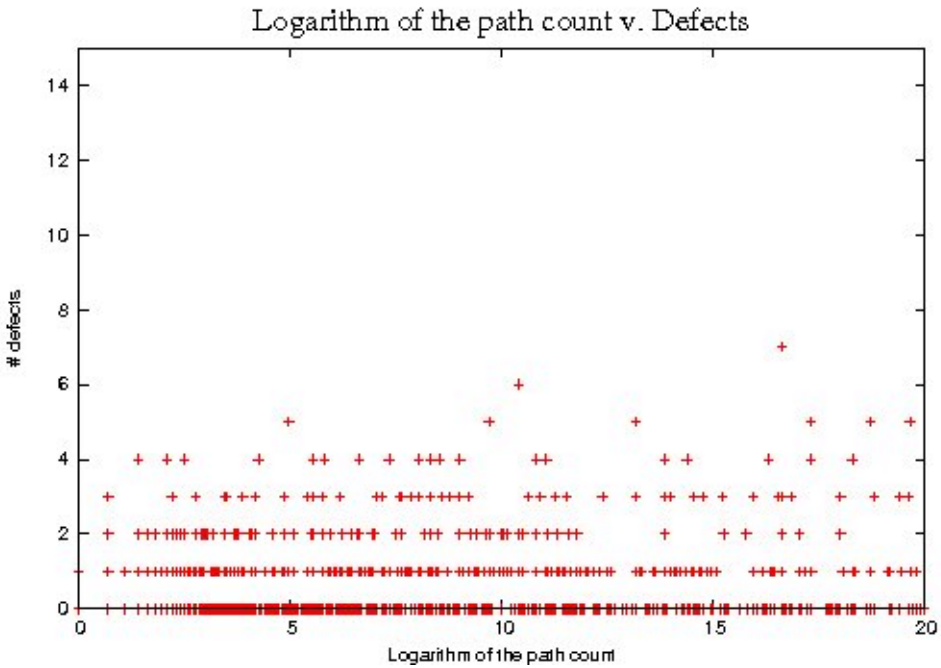


knots v. defects

Defect correlations

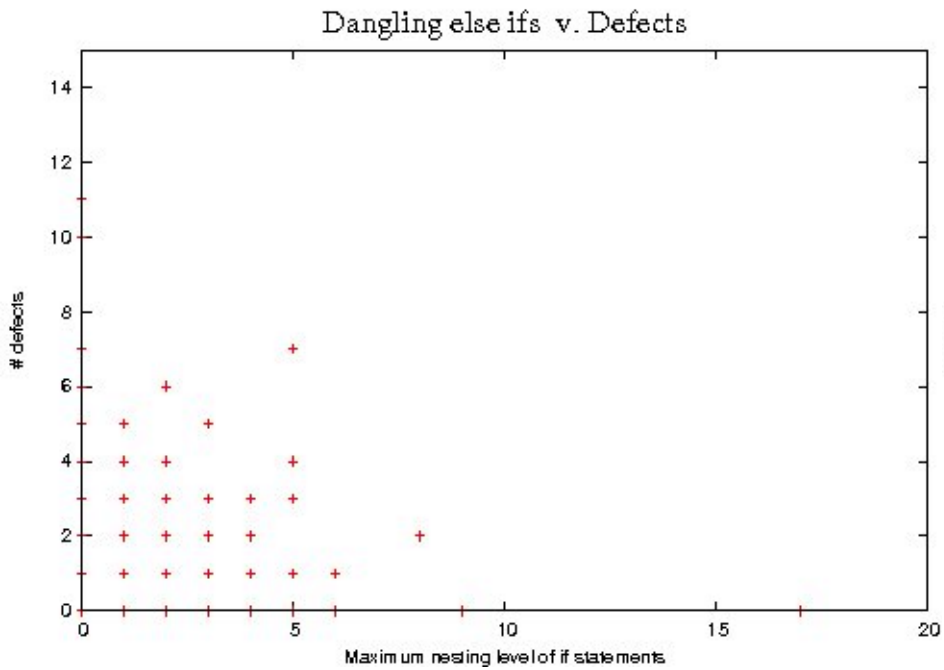


Shortest jump v. defects

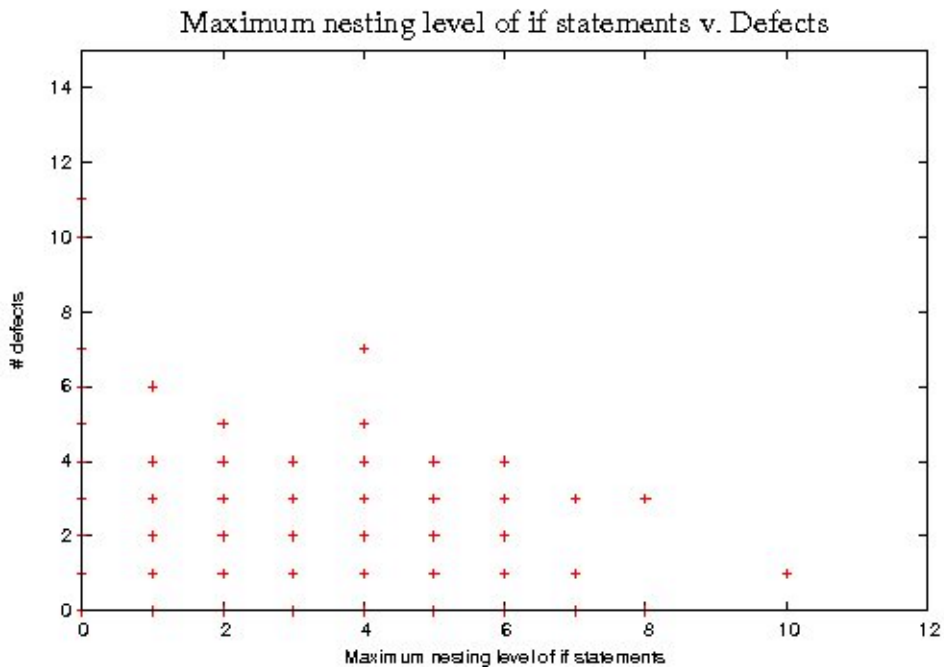


Ln(path count) v. defects

Defect correlations

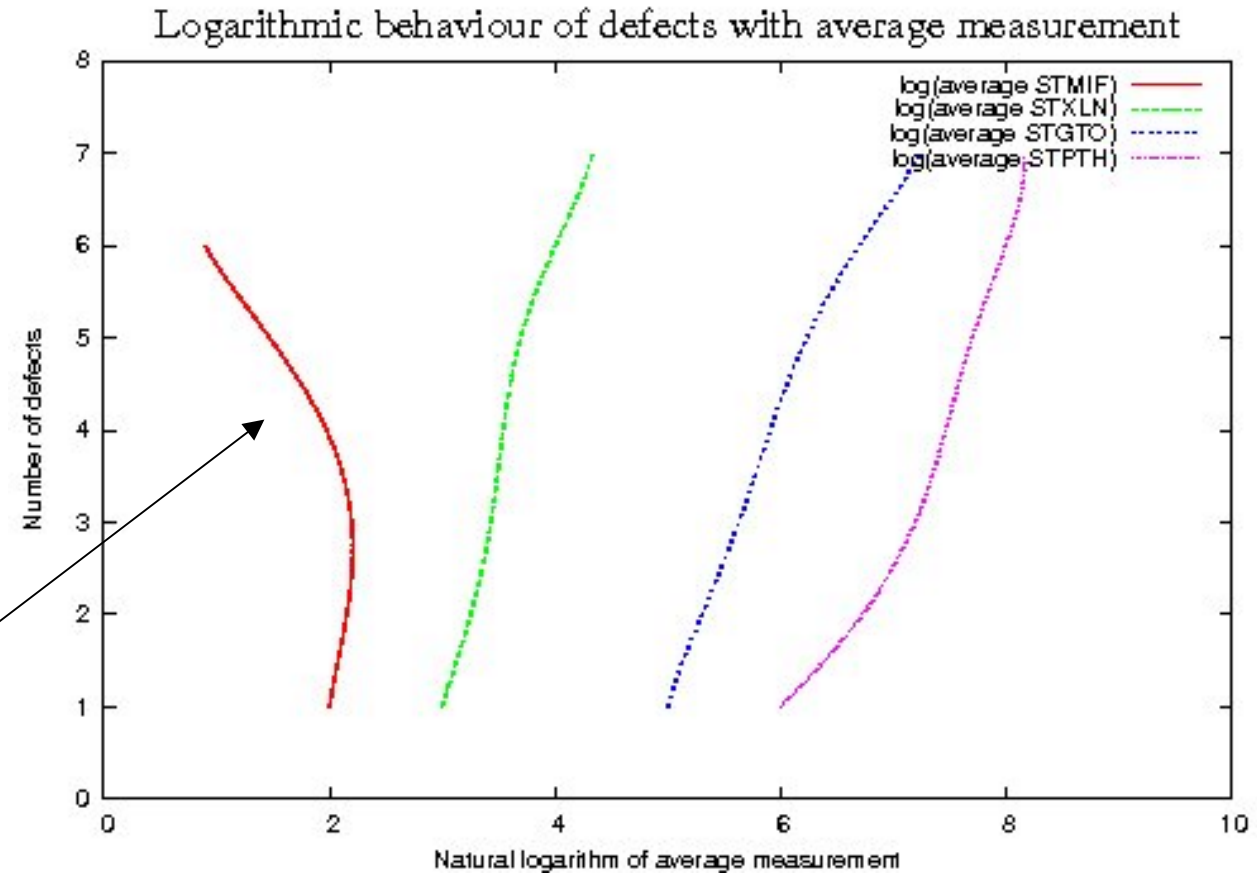


Dangling else if v. defects



Maximum if depth v. defects

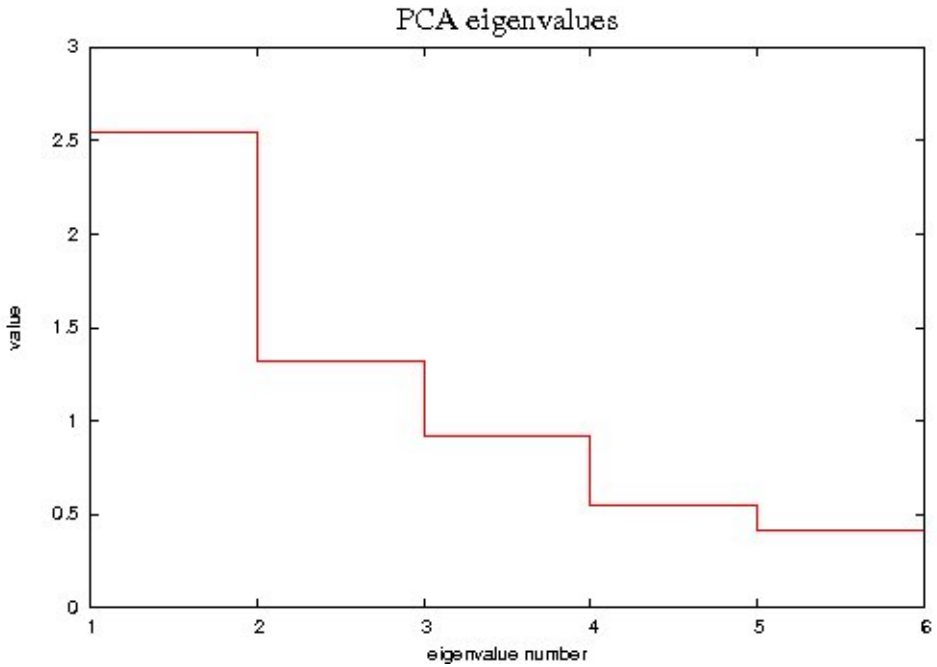
Defect correlations averaging over component ranges



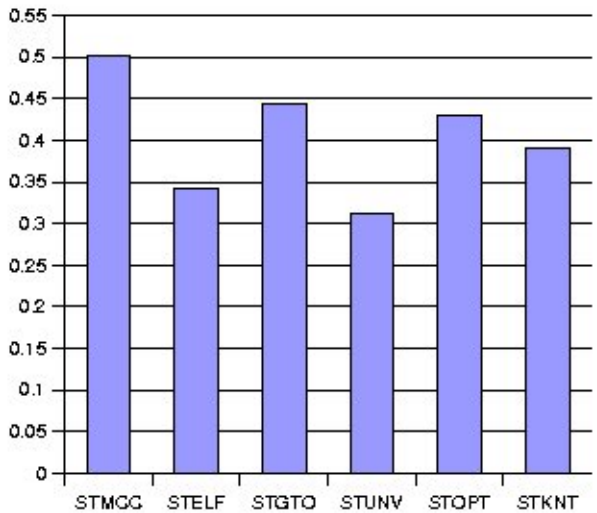
Note

Log (averaged metrics)

PCA

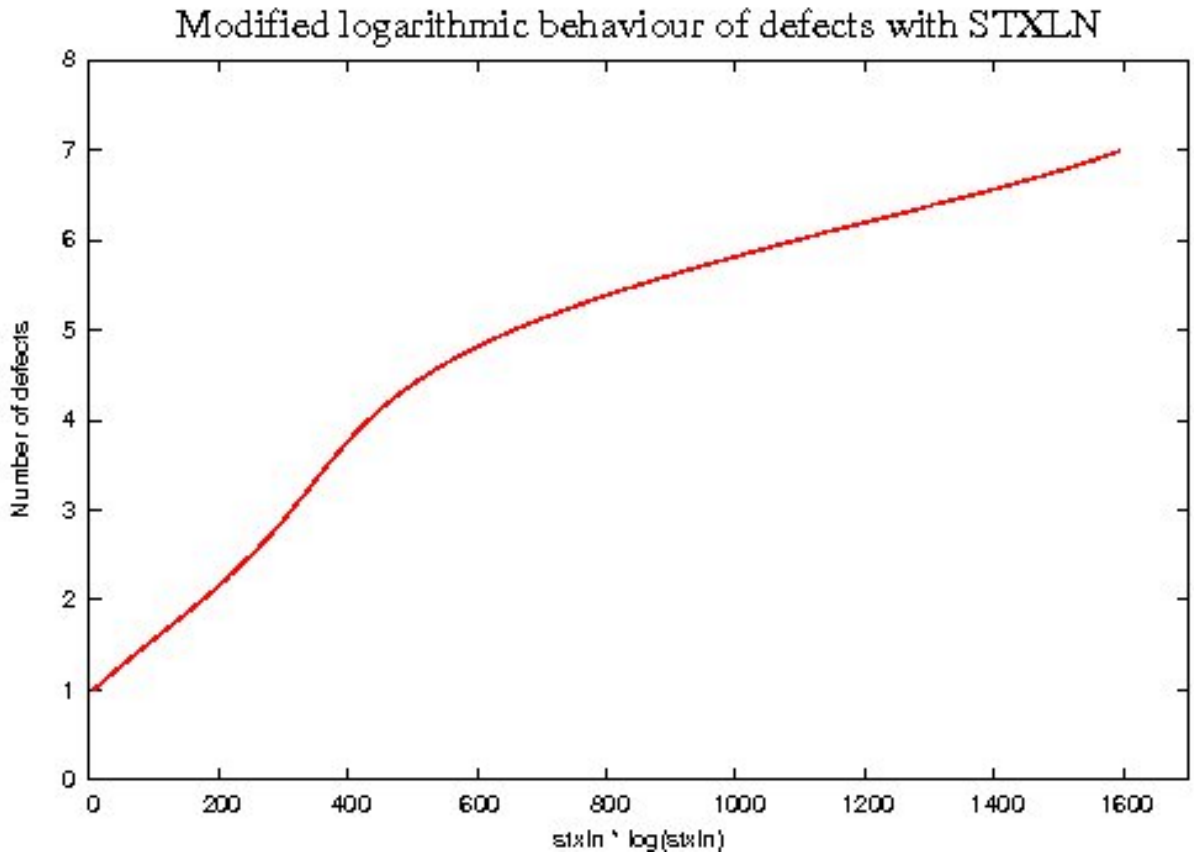


Dominant eigenvalues



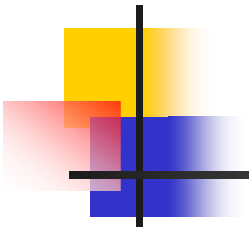
Principle eigenvector

More evidence of scale-free behaviour



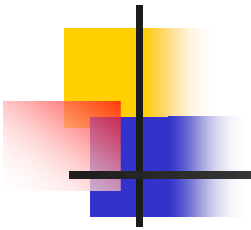
Zones of linearity in xloc ln(xloc)

Overview



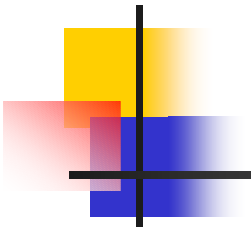
- Some background
- The NAG Fortran library
- Defect analysis
- Conclusions

Conclusions



- So-called software metrics are highly correlated here
- PCA reveals that there is no simple relationship between defects and structural “metrics”
- The goto statement is not particularly harmful
- The NAG library also exhibits scale-free behaviour.

References



My writing site:-

http://www.leshatton.org/