Pipelines of 'Wells' in Mathematics and Computer Science

E. Anderson[1,2] **and A. Ford**[1]

Abstract

We introduce the conceptualization of parts of Mathematics and Computer Science as pipelines of soundness gates. The passing of each of which is coordinatedly named by a word starting with 'well'. Sequels shall cover various AI Age applications of this.

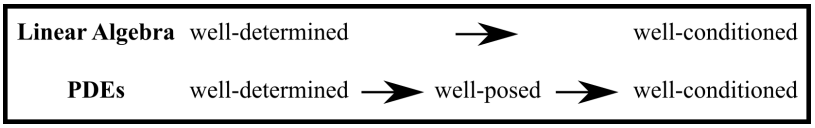

Figure 1:

This article is ⟨3⟩ : understandable by third-year undergraduates.

Cite as E. Anderson and A. Ford "Online Encyclopaedia of Applied Topology" institute-theory-stem.org/pipelines-of-wells/ (2026).

1 Determinedness in Linear Algebra

Definition 0 A Linear-Algebraic system is *well-determined* if its number of unknowns matches its number of equations. Elsewise, it is *ill-determined*. Binarily splitting into the following qualitatively distinct types. *Over-determined* if it has more equations than unknowns. And *under-determined* if vice versa.

Remark 1 This example is widely-known, even from vectors and matrices before Linear Algebra enters consideration. As a count (mis)match, this transcends to whichever combination of non-linear and not merely algebraic systems.

2 Wells and ills of Mathematical Physics

2.1 Posedness

Definition 1 A *differential equation problem* is a system of differential conditions and some prescribed boundary data. Defined generally enough to include totally or partly temporal boundary data. A differential equation problem is *well-posed* [2, 5] if the following holds. The solution exists, is unique and depends continuously on the boundary data.

Remark 1 The above is to be viewed as a minimal set of well-posedness criteria. That specific classes of differential equation systems are well capable of adding extra criteria to. For instance hyperbolic partial differential equations encode finite propagation speeds. And consequently require a causal extension of well-posedness to include the domain of dependence property [2, 3].

Definition 1′ A differential equation problem that is not well-posed is *ill-posed* [4].

Remark 2 Types of ill-posed are far more conceptually diverse and numerous [4] than the ill-determined binary.

[1]Institute for the Theory of STEM Date stamp: v1: 24-11-2026.

[2] Corresponding Author dr.e.anderson *at* maths.physics.protonmail.com. Copyright of Dr E. Anderson.

Remark 3 Undergraduate STEM courses essentially always only dish out well-posed problems. Theoreticians, and even Nature Herself, however, do not in general ascribe to this nicety. Even many theoreticians are not aware that the general situation consists of a mixture of ill-posed problems and problems not known to be well-posed alongside the standard well-posed problems. The Mathematical subject called Analysis [8] was invented so as to render commonly encountered PDE mathematics rigorous. Standard Analysis often does not extend to further types of PDE system. Or to the same PDE system with different kinds of initial/boundary data, which is what posedness refers to. Let this paragraph serve as an example that many researchers remain unaware of some parts of pipelines, even in their own subjects.

2.2 Conditionedness

Definition 1 A system of equations is "well conditioned" [5] if small perturbations in the input result in small changes in the output.

Remark 1 Well-conditioning already applies in basic Algebra and Geometry. Division by a small number, and finding the intersection point of a pair of almost-parallel lines are examples.

Remark 2 One general technique for resolving ill-conditioning is to add artificial penalty terms.

Remark 3 A further subtlety at the level of PDEs from gauge theories is as follows. Two formulations of a PDE system can be Mathematically equivalent. And yet the first is well-conditioned while the second is ill-conditioned. As is evidenced by computer codes crashing for the second but not for the first.

Remark 4 A given gauge theory, of which General Relativity is a large example, does not however tolerate Remark 2's alterations to its laws. One may however be able to judiciously pick a gauge that avoids ill-conditioning. For instance, one might be able to find another gauge in which the error terms propagate away from the location being investigated. As opposed to remaining there and blowing up. This phenomenon became known to one of us in the context of the Numerical General Relativity of colliding pairs of black holes [9].

Remark 5 So far, we can assemble the Linear Algebra and Mathematical Physics PDE system pipelines in Fig 1.a)-b).

3 More foundational layers of understanding

Remark 1 Some further 'wells' preceding the above pipelines are as follows.

3.1 Definitions

Definition 1 A notion is *well-defined* if it is independent of presentation.

Example 1 Say independent of how a graph is drawn, or a gauge-independent definition in the context of a Gauge Theory.

3.2 Notation

Definition 2 An object is *well-denoted* if no other object in that treatise is allocated the same symbol. Or possibly even a reasonably confuseable symbol.

Remark 1 Using a recognizable symbol is furthermore helpful. Some very commonly encountered objects, such as nabla, may break out of this Remark's rule of thumb.

Remark 2 In contrast, ill-denotation overloads symbols. Observe that this occurs for some unit of text's collective notation rather than for any one object's notation in isolation.

3.3 Naming

Definition 3 An object is *well-named* if the name accurately describes some combination of its structure, function or origin.

Remark 1 False analogues are to be avoided. So are excessively long names.

3.4 And back to notation

Definition 2′ An object is *mimetically well-denoted* if its notation resembles its name.

Example 1 *Leading-letter well-denoted.* For instance summation is denoted by a capital sigma and products by a capital pi.

Example 2 *Iconic well-denoted.* Also the ceiling and floor functions' brackets have top- and bottom-end bends away from their vertical lines. Corresponding to the ends at which ceilings and floors are to be found relative to walls.

Remark 3 A truer-name upgrade often induces a truer-notation upgrade of some mimetic type.

3.5 Conceptualization

Definition 4 A clearly thought-out notion is *individually well-conceptualized.*

It is *collectively well-conceptualized* if related concepts (variants, opposites, foils, combinations) have been assessed for completeness using the Wille concept lattice [7].

A further criterion, after this collective analysis, is whether the initial concept remains optimal for its initially intended purpose. And whether it maintains *Leibnizian sufficient reason* for consideration.

A further a posteriori criterion of well-conceptualization is that a Mathematically sharply-defined version turns out to be useful in a wide range of contexts.

Example 1 Compactness is a well-known example.

Remark 1 With the intended applications at hand being various aspects of the AI Age, the Author of everything so far (E.A.) joined forces with a Theoretical Computer Scientist (A.F.). This resulted in a few modifications to the basics – Linear Algebra – and to the overlap: conditioning. It also became profoundly clear that AI Age pipelines are much closer to Computer Science's. Both in sheer number of soundness gates and as regards need and willingness to identify more, make finer conceptual distinctions and coin names.

3.6 Axiomatization

Remark 1 Let us firstly recollect that Mathematical axiomatization is ambiguous. That some are inconvenient, so one might for instance *well-axiomatize* so as to avoid Russell's paradox. And that some are overly simple, such as ones that avoid Gödel incompleteness by not even managing to house Arithmetic.

4 Computer Science's wells

4.1 Formal Logic and Programming Language syntax

Definition 1 Syntax is *well-formed* [6, 10] if it conforms to formal grammatical rules.

Definition 2 Syntax is *well-typed* [6] if its expressions satisfy some prescribed static set of type-checking constraints.

Definition 3 Syntax is *well-founded* if its recursions and inductive orders finitely terminate.

Definition 4 Syntax is *well-structured* if its sequence, selection and iteration constructs are clean.

Definition 5 Syntax is *well-scoped* if its variable bindings are unambiguously resolved within some lexical or dynamic environment. Where bindings refers to identifiers as handles for some underlying environment's entities. Such as memory locations, values, functions or types.

Definition 6 Syntax is *well-bracketed* if its delimiter nesting is balanced across each expression or blocks.

Definition 7 Coding syntax is *well-parsed* from the Human-facing point of view if the following holds. It is written in monospaced font. With open and closed bracket pairings vertically stacked. And item nesting picked out by horizontal tabbing.

Remark 1 Syntax is another place where technical well-behavedness criteria need to be unpacked.

4.2 Data Structures

Definition 1 A function is *well-defined* if it produces unique output independent of representation.

Definition 2 A search tree is *well-balanced* if it adheres to logarithmic height bounds across deletions and insertions.

Definition 3 A data structure modelled by a graph is *well-connected* if the following holds. Its vertex and edge connectivities are robust against fragmentation into components.

Definition 4 A date structure (or a network) is *well-partitioned* if it is split up so as to minimize cross-node communication.

Remark 1 This is another place where technical well-behavedness needs to be unpacked. Here that functions are to operate within predictable complexity and convergence bounds. Algorithms are to operate likewise too.

Remark 2 The above well-defined is in the sense used in Sec 3.1. In Numerical Analysis extensions of the current subsection's material, the above definitions of well-posed and well-conditioned are relevant. Common types of well-conditioned here include floating-point insensitivity. And well-scaling, leading to well-flowing numerics. As opposed to the binary of under- and over-flowing numerics.

4.3 Software Architecture and Engineering

Definition 1 A system is *well-architected* if it has the following attributes.

'Floorplans': clear boundary delineations between components.

And 'zooming': scaleable dependency hierarchies wherever relevant.

Definition 2 A codebase is *well-modularized* if it is partitioned into cohesive, substantially decoupled units.

Definition 3 Software is *well-engineered* if maintainable abstractions, fault isolation, and defensive invariants against error are baked in.

Definition 4 Individual modules are *well-encapsulated* if they conceal their internal states behind explicit interface contracts.

4.4 Extension to verification and distributed systems

Definition 1 A computer program is *well-specified* if governed by formal pre- and post-conditions, and invariants.

Definition 2 It is *well-verified* if Mathematically proven to satisfy given formal specifications.

Definition 3 It is *well-tested* if empirically validated over a comprehensive set of tests.

Definition 4 It is *well-synchronized* if concurrently operating threads enjoy coordinated shared-state access without deadlocks.

Definition 5 It (or a codebase) is *well-documented* if accompanied by accurate, synchronized Human-readable explanations.

Definition 6 Computing systems or infrastructure are *well-provisioned* if the allocated resources are not exceeded by the peak capacity.

4.5 Negation in Computer Science

Remark 1 In Mathematics, the obstruction at a soundness gate is accorded a word starting with 'ill'. In contrast, in Computer Science, the 'non' and 'un' prefixes are rather more common than 'ill'.

References

[1] A.M. Turing, "Rounding-Off Errors in Matrix Processes" Quart. J. Mech. Appl. Math. **1** 287 (1948).

[2] R. Courant and D. Hilbert, *Methods of Mathematical Physics* **2** (Wiley, 1962).

[3] R.M. Wald, *General Relativity* (U. Chicago P., Chicago, 1984).

[4] Karen A. Ames and B. Straughan Book, *Non-Standard and Improperly Posed Problems* (AP, 1997).

[5] L.N. Trefethen and D. Bau, *Numerical Linear Algebra* (SIAM, 1997).

[6] B. Pierce, *Types and Programming Languages* (MITP, 2002).

[7] B.A. Davey and Hilary A. Priestley, *Introduction to Lattices and Order* (C.U.P, Cambridge 2002).

[8] T.W. Körner, *A Companion to Analysis: A Second First and First Second Course in Analysis* (C.U.P., Cambridge 2004).

[9] O. Sarbach and M. Tiglio, "Continuum and Discrete Initial-Boundary Value Problems and Einstein's Field Equations" (Living Reviews in Relativity **15** 9 (2012).

[10] R. Harper, *Practical Foundations for Programming Languages*, (CUP, Cambridge 2012).

[11] A. Timany, R. Krebbers, D. Dreyer and L. Birkedal, "A Logical Approach to Type Soundness", J. ACM **71** 40 (2024).