COMPUTATIONAL MATHEMATICS IN COMPUTER ASSISTED PROOFS

1. Workshop Summary

Computer assisted proofs are becoming a mainstay in modern mathematics, as there are an increasing number of famous conjectures and theorems that have recently been proven using computer assisted proofs, where the mathematical areas include packing problems, low dimensional manifolds, number theory, group theory, spectral problems and mathematical physics, differential equations etc. A highly incomplete list in alphabetical order includes the Dirac-Schwinger conjecture, the Double-Bubble conjecture, Kepler's conjecture (Hilbert's 18th problem), Smale's 14th problem, the 290-theorem, the weak Goldbach conjecture etc. In many of these cases the proofs are based on using numerical computations with approximations, a technique that belongs to the field of computational mathematics. The workshop focused on the interplay between computational mathematics and pure mathematics as numerical approximations used in computer assisted proofs may be an incredibly powerful tool for pure mathematicians and such tools may change the way we approach many problems in mathematics. However, in order to unleash the full potential of computational mathematics in computer assisted proofs there are substantial mathematical hurdles and also challenges regarding how to make the two usually quite disjoint communities of pure and computational mathematicians work together. This workshop was an initial step.

1.1. **Ten Talks: From PDEs, topology and number theory to foundations of computations and AI.** The following talk were presented.

(i) (Monday) "*Computer assisted proofs for blowup solutions of 3D Euler equations and related models*" (Tom Hou, Caltech),

(ii) (Monday) "*The Solvability Complexity Index hierarchy & generalized hardness of approximation – On the role of foundations of computation in computer assisted proofs* " (Alex Bastounis, University of Edinburgh/Matt Colbrook, Ecole Normale Superieure, Paris and University of Cambridge),

(iii) (Tuesday) "*Computational tools and proofs in low-dimensional topology and geometry*" (Andrew Yarmola, Princeton University),

(iv) (Tuesday) "*Validated numerics - Some algorithms and practical applications*" (Mioara Joldes, CNRS),

(v) (Wednesday) "*The future of computer assisted proofs*" (Kevin Buzzard, Imperial College),

(vi) (Wednesday) "*Determining rational points on curves: computational challenges and reproducibility*" (Jennifer Balakrishnan, Boston University),

(vii) (Thursday) "*Deep learning: Mathematical framework for understanding it*" (Sanjeev Arora, Princeton Univerity),

(viii) (Thursday) "*Rigorous computations, (a little bit of) automated proving and (hopes for) formal proofs: a weak-Goldbach miscellany*" (Herald Helfgott, University of Göttingen/CNRS),

(ix) (Friday) "*AI-powered mathematical discovery*" (Petar Velickovic, DeepMind),

(x) (Friday) "*Computer-assisted proofs in PDE*" (Javier Gomez-Serrano, Brown University).

2. Problem Sessions

The problems for the problem sessions were suggested by the participants and then voted on in order to make the final list below. The ordering of the list below is based on the chronological order in which the problems were suggested.

Problem 1: Are computer-assisted proofs real proofs? (K. Buzzard). *Are computer-assisted proofs always real proofs when proprietary software and/or hardware is involved that is not accessible to all readers/reviewers?*

This can be viewed as a philosophical question, but there is also the practical question of whether computer-assisted proofs are acceptable to the mathematical community, and there is clear evidence that nowadays they are; things have moved on a lot from the situation that Appel and Haken found themselves in in the 1970s, when people were less familiar with computers. Discussions were had about how to reduce reliance on software/hardware. There are two distinct issues here. Firstly, can one move computations from closed-source systems such as magma to open-source systems such as sage-math (answer: yes, but it involves work and there is not much incentive), and secondly, can one move computations from unverified systems such as sage-math to verified systems such as Lean (answer yes in theory, but it involves a huge amount of work in practice and there is again not much incentive).

Problem 2: What is the right metric to determine computability? (T. Hou). *In problems where mathematical analysis precedes a computer assisted step: How can we ensure that the formulation of problem is correctly posed so that computability and non-computability of a problem can be determined?*

The discussions focused on the stability versus instability of a potential finite time singularity of a nonlinear PDE or ODE. We used several examples from nonlinear PDEs and dynamic systems to illustrate the importance of the the appropriate functional space and the associated norm (or metric) in studying the stability or instability of a potential blowup solution. We spent quite some time discussing a recent paper entitled "Blowup solutions to 3D Euler are hydrodynamically unstable" by Vasseur and Vishik (CMP, 2020). Vasseur and Vishik studied the instability of the Euler blowup based on the growth estimate in L^p norm of the linearized Euler equation around a blowup solution. While such result is mathematically correct, the linearized Euler equation around a blowup solution will necessarily blow up at the same time as the background blowup solution. It does not capture the change of blowup time, the blowup exponent and the blowup profile due to the change of the initial data. One can easily show that such definition of instability would also imply that the blowup of a Riccati ODE or the invicid Burgers equation is unstable. On the other hand, using a modulation technique or a dynamic rescaling formulation, one can easily show that both the Riccati ODE and the inviscid Burgers equation have stable self-similar blowup solutions. Recently, Chen and Hou showed that the method of analysis by Vasseur and Vshik can be generalized to prove that the finite time blowup solution of the 3D axisymmetric Euler equation and the 2D Boussinesq equation with $C^{1,\alpha}$ initial velocity is also hydrodynamically unstable using the definition of instability introduced by Vasseur and Vishik. However, Chen and Hou already proved that the 3D axisymmetric Euler equation and the 2D Boussinesq equation with $C^{1,\alpha}$ initial velocity have a stable finite time self-similar blowup using the dynamic rescaling formulation (CMP 2021). The stability obtained by Chen-Hou is the stability of the self-similar profile, which naturally accounts for the change of the blowup time, blowup exponent and blowup profile due to the change of the initial data. The take-away message is that the question of stability or instability depends crucially on the choice the functional space and the metric that we use to study stability. A blowup solution could be stable in one functional space but is unstable in another functional space at the same time. The main question is to look for an appropriate formulation and the appropriate functional space to study the stability or instability of a potential blowup solution. It could be misleading to conclude that the 3D Euler blowup is not computable based on one formulation and one metric that are not suitable to study the potential stable blowup of the 3D Euler equation.

Problem 3: Create guaranteed eigenvalue solver for matrices (S. Olver). *Given a $n \times n$ matrix, can you output the spectrum/eigenvalues to given accuracy? - In polynomial time. Create software that can do it with 100% guarantee.*

In our discussion it was pointed out that the state-of-the-art was a lot further than what originally thought, in particular Arb can compute spectrum of dense matrices using high-precision interval arithmetic in a validated scheme (that is, where a non-rigorous method gives initial approximations that are then validated with interval arithmetic). Arb was amazingly robust in numerical tests, achieving bounds on the spectrum on the order of 10^{-70}. There is still a lot to do on this problem:

1. 100% guarantee apriori. Arb is already guaranteed to be correct when it succeeds but may fail on specific matrices.
2. Rigorous proofs. E.g. Arb could potentially generate a computer verifiable proof but at the moment this is not done.
3. Support for sparse or otherwise structured matrices. Here there seemed to be an issue that one needs to construct inverses of the resolvent in order to do the verification process, thus losing the sparsity. Possibly banded matrices could be handled using a semi-separable representation for the inverse.
4. High performance. The same techniques done in Arb using high-precision arithmetic could be done in floating point arithmetic with a significant speed-up.

Problem 4: Packing problems in geometry (J. Lagarias). *Find a dimension where the densest sphere packing is not a lattice. How many regular tetraheadra in $\mathbb{R}^3$ can touch at a single point?*

The workshop discussions covered several problems on packings. Such problems lead to large, hard computational problems.

(Q1) Find a dimension where the densest sphere packing is not a lattice.

Ideas. According to Conway-Sloane data, the smallest dimension where the current best packing is not lattice is $n = 10$ (it is several cosets of a lattice). There are many other dimensions up to $n = 22$ where the currest best packing is not lattice. To prove the $n = 10$ case best packing is not lattice, it suffices to bound the densest lattice packing below that of current best packing. There is an effectively computable algorithm that finds all the best lattice packings in dimension n. It has run up to dimension 8. Dimension 9 is under investigation and is very large computation, but according to Henry Cohn. the computation is almost finished. Dimension 10 is completely out of reach for this method, too large a problem. It would be nice to find another method that can get a provable upper bound for densest lattice packing without treating the general case.

(Q2) (Henry Cohn) Treat the problem of Gaussian core packings in dimension 3. with parameter $\lambda = 1$.

This problem was introduced by Frank Stillinger [J. Chem Phys **65** (1976), 3968–3984. The energy function between two sphere centers x and y in $\mathbf{R}^3$ is $\exp(-\lambda||x - y||^2)$. The question is what is the total energy minimizer, and is achieved by a lattice packing. The answer should be "no" because at $\lambda = 1$ the Face-Centered cubic lattice (FCC) and its dual lattice (FCC*) both attain an equal energy, their energy curves cross at this point. So a material with a mixture of the two phases should have lower energy. It cannot be periodic, there will be bigger and bigger pieces of the two lattices, with boundaries where they don't meet perfectly. These boundaries are 2-dimensional so their effect attenuates on bigger and bigger regions. To prove lattice packings cannot minimize this energy one has to estimate across all 3-dimensional lattices. The minimal distance vector for FCC and FCC* packings of volume 1 were determined. They are $2^{-1/6}$ for FCC and the sixth root of $27/32$ times $2^{1/6}$ for FCC*. Andrew Yarmola thought he was getting somewhere computationally on the problem, since the energy is a theta function for a lattice, and one wants to study the space of theta functions of 3-dimensional lattices of a fixed determinant. He hoped to show FCC and FCC* were the minima.

(Q3) What is the maximum number of congruent regular tetrahedra that can touch at a common point (a tip of a vertex for all of them.) [Stated in paper of Lagarias and Zong, Notices AMS. But the problem was around before that.]

It is known 20 tetrahedra touching is attainable, volume considerations allow 22.8, so 22 at most is possible. This is a large computation, not clear how to formulate it. The problem can be reduced one dimension, intersect with a small sphere centered at the point, get equilateral spherical triangles with angle $arccos(1/3)$. How many can one pack of these on a 2-sphere? Joel Hass suggested trying to eliminate one case, the "easiest" case, can 11 such spherical caps fit without overlap on real projective space, which is S^2 quotiented by an involution through the center point. That is, the problem is to pack 22 equilateral spherical caps, with the angles given above, which form a set of 11 caps are antipodal to the other 11. No clear progress was made on this problem; it does seem to be the most approachable of the problems of this kind, the hope being to prove it is impossible.

Noam Elkies suggested "unwrapping the surface of an icosahedron on the 2-sphere corresponding to packing 20 (like an apple peel) into chains of connected triangles, and seeing if any of them left room to fit in another equilateral spherical triangular cap. He said there were about 88000 possible unwrappings.

(Q3a) Lagarias pointed out one can generalize the problem above, to that of packing of identical spherical triangles of angle θ on the 2-sphere, how much uncovered area (out of 4π) would there be? (it depends on θ.

The angle varies betwen $\pi/3$ (Euclidean limit) where the triangles get very small and 2π (where the triangle gets large). Perfect tilings exist for $\pi/2$ (octahedron), $2\pi/3$ (octahedron) and $2\pi/5$ (icosahedron). What is the "worst" value of θ giving the least dense packing? No clear progress on solving any case.

(Q4) Joel Hass raised the problem of classifying all the (congruent) tetrahedral shapes that can tile space.

This problem has a long history. Two parametric families of shapes that tile are known and some additional isolated cases. According to Hass, after removing homotheties by fixing the volume of the tetrahedron, there is a 5-dimensional (real) manifold of shapes. Dehn invariant is 0 is one constraint, but this is a difficult invariant, this set may be dense in the manifold. The guess is that the answer will be a family of curves on the manifold plus some isolated points. There is an article on this problem by Marjorie Senechal in Math. Magazine in 1981. No ideas were reported on this problem.

Of the problems discussed, the most progress was reported on the problem that Henry Cohn reported on the Gaussian core packings in 3-dimensions, with parameter $\lambda = 1$.

Problem 5: Phase transitions and the extended Smale's 9th problem (J. Lagarias). *Given the new results on the extended Smale's 9th problem, what are the right input sets to obtain a polynomial time algorithm in the extended model for linear programs (LPs)? Exhibit a problem that has* 3 *(or more) phase transitions in generalised hardness of approximation for LPs – for example one with a transition from polynomial to exponential time – only* 2 *phase transitions are known so far. Does there exist an algorithm for LPs that is polynomial time (in arithmetic operations) independent of the bit-size of the input, and if so can this lead to sharper bounds on particular phase transitions in the extended model?*

The workshop discussion concentrated on two aspects of the suggested problems above.

(Q1) The phase transitions established in the results on the extended Smale's 9th problem exhibit the phenomenon of generalised hardness of approximation. It is natural to ask how to understand this phenomenon in view of the above problems.

Shmuel Weinberger pointed out the delicate issue of characterising the different phase transitions in terms of specific conditions. For example, it is somewhat paradoxical that the currently known phase transitions are independent of the many condition numbers in optimisation. Thus, finding sufficient and necessary conditions for phase transitions may shed light on the intricate phenomenon of generalised hardness of approximation.

This problem is directly related to Lagarias' questions above, but hard to solve as the conditions will vary with the different types of phase transitions. Another problem that was discussed was the following: how many phase transitions can exist for LPs? So far two phase transitions are known, but establishing an upper bound on the number of different transitions is an unsolved problem. It seems likely that one can find three different phase transitions expanding on existing techniques, but how to obtain more than three is certainly not clear.

Charlie Fefferman raised the following point during the presentation on the extended Smale's 9th: Given that one can, for any $K \in \mathbb{N}$, obtain sets of LPs in the extended model for which one can compute K correct digits of a minimiser with 2/3 probability if the randomised algorithm has a non-zero probability of not halting, but no random algorithm can compute K correct digits with probability $> 2/3$, this suggests the existence of a collection of particularly difficult LPs. In particular, the standard technique of rerunning a '$2/3$-algorithm' will not improve the probability of success. How can these particularly difficult LPs be characterised? Fefferman's point is subtle, as the results on the extended Smale's 9th imply the existence of a collection of LPs for which there are cases where any randomised algorithm – that can attain $2/3$ probability of success – will have a probability of running forever equal to $1/3$. This explains why rerunning a '$2/3$-algorithm' will not work. However, a characterisation of these difficult LPs would be very interesting, and this is not currently available.

Ilse Ipsen raised the issue with finite precision. Algorithms running with finite precision may in practice exhibit worse phase transitions than what the theory predicts, as the theory demonstrates phase transitions even when infinite memory is assumed. Moreover, different models of rounding may affect the observed phase transitions, however, the exact impact on the phase transitions when moving from arbitrary precision to finite precision computation in LP is currently unknown.

(Q2) The link between the extended Smale's 9th problem and the Solvability Complexity Index (SCI) hierarchy.

There is a fundamental link between the extended Smale's 9th problem and the SCI hierarchy. Firstly, the results on phase transitions implicitly imply phase transitions changing between classes in the SCI hierarchy. Secondly, phase transitions due to generalised hardness of approximation typically imply non-computability, however, this does not necessarily mean that the computational problem cannot be used in a computer assisted proofs. The key issue is that as long as the computational problem is in either Σ_1 or Π_1 in the SCI hierarchy, it can still be used in a computer assisted proof.

David Gabai pointed out how the computational problems used in many of his computer assisted proofs in low-dimensional topology can be viewed in terms of classifications in the SCI hierarchy. The follow up discussion focused on how the scope of the SCI hierarchy extends across the whole of computer assisted proofs and how many computer assisted proofs implicitly prove classification results in the SCI hierarchy. This is for example the case in the computer assisted proof of the Dirac-Schwinger conjecture by Fefferman and Seco.

Problem 6: The water wave graph to splash (C. Fefferman). *The water wave equations (or 2D incompressible free boundary Euler equations) describe a system consisting of a water region $\Omega(t) \subset \mathbb{R}^2$ and a vacuum region $\mathbb{R}^2 \setminus \Omega(t)$, evolving as a function of time t, and separated by a smooth interface. The fluid is assumed to be incompressible and irrotational and to satisfy the 2D Euler equation under gravitational forces. We assume that the pressure at the interface is only a function of t and that the interface moves with the fluid. The simulations (Fig. W) show an initially smooth water wave, for which the fluid interface is a graph as in Figure X. At a later time t_1, the water wave has 'turned over', i.e., the interface is no longer a graph. Finally, in Figure Z, the fluid interface self-intersects at a single point, but is otherwise smooth. We call this scenario a 'splash', and we call the single point at which the interface self-intersects, the 'splash point'. Castro–Córdoba–Fefferman–Gancedo–López-Fernández [1,2] already proved that a water wave may start as*

in Figure X and later evolve to look like Figure Y. Castro–Córdoba–Fefferman–Gancedo–Gómez-Serrano [3] proved that a water wave may start as in Figure Y, and later form a splash, as in Figure Z. The open problem is to prove that an initially smooth water wave may start as in Figure X, then turn over as in Figure Y, and finally produce a splash as in Figure Z.

[1] A. Castro, D. Córdoba, C. Fefferman, F. Gancedo, and M. López-Fernández. Turning waves and breakdown for incompressible flows. PNAS (2011).

[2] A. Castro, D. Córdoba, C. Fefferman, F. Gancedo, and M. López-Fernández. Rayleigh-taylor breakdown for the muskat problem with applications to water waves. Ann. of Math. (2012).

[3] A. Castro, D. Córdoba, C. Fefferman, F. Gancedo, and J. Gómez-Serrano. Finite time singularities for the free boundary incompressible Euler equations. Ann. of Math. (2013).

During the discussion, several ideas were presented. A promising approach was to consider the linear operator and extract the leading order terms out of it, since previous approaches [4] didn't take advantage of them and only did crude bounds. Building up on that splitting, it may be useful to precondition the system by premultiplying by an inverse or an approximate inverse of the linear operator. Other potential ideas were to fabricate a scenario which holds true at an asymptotic level in time (i.e. graphs that go to a splash arbitrarily quickly, under proper normalization conditions) to treat the problem using perturbation techniques.

[4] J. Gómez-Serrano. Analytical and computer-assisted proofs in fluid mechanics. PhD Thesis. Universidad Autónoma de Madrid, 2013.

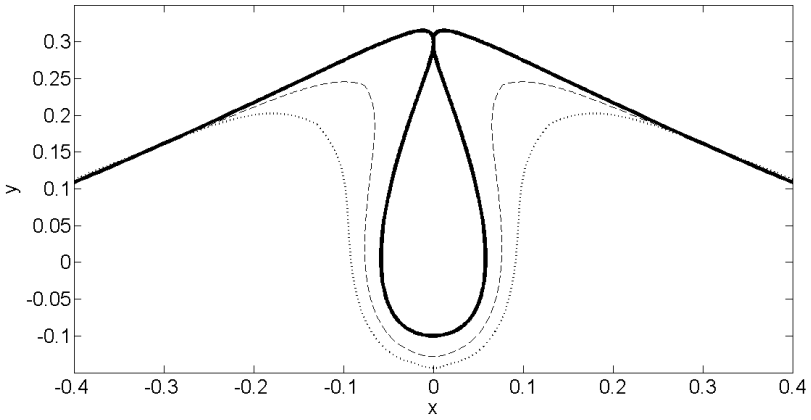
Problem 7: Validated numerics (F. Brehard). *Validated numerics in computer assisted proofs.*

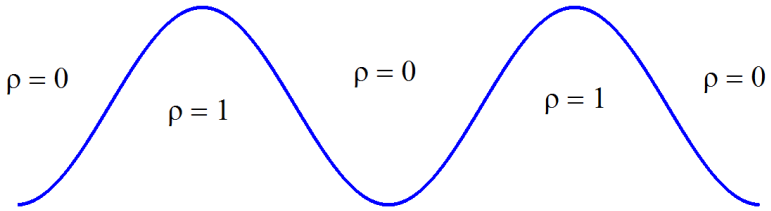
The group was composed of developers and (actual or potential) users of numerical tools for computer-assisted proofs. They discussed about various existing software tools ranging from numerical (ApproxFun.jl, CR-Libm) to validated (Arb, MPFI, ValidatedNumerics.jl, CAPD) or certified (Gappa, CoqInterval). Sharing their own experience in this domain, they sketched out some of the desired features of tomorrow's ideal validated numerics software for mathematicians. For instance, a roadmap for such a software could consist of: (1) an efficient core library written in a universal language such as C; (2) an easy interface (e.g., a wrapper in Python or Julia); (3) a proof assistant- side certification which would rule out possible bugs in the implementation.

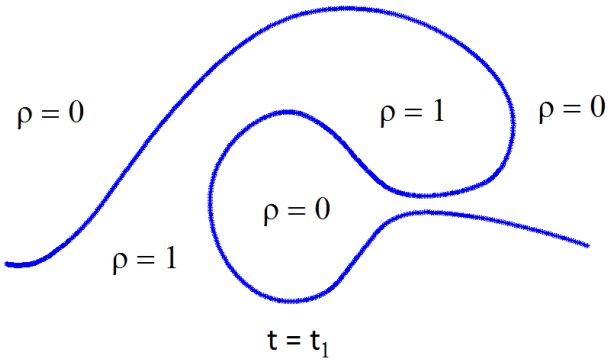
Problem 8: AI in computer assisted proofs (A. Hansen). *How to use AI in computer assisted proofs and computer assisted intuition?*

The discussion revolved around several aspects of AI in computer assisted proofs. Specifically, the group focused on two particular issues: (1) the use of AI techniques to provide an 'initial guess' for a candidate object that can later be formally verified by a checker algorithm verifying that the object satisfies the desired requirements needed for the proof to be valid; (2) the use of AI techniques to assist the mathematician developing intuition about the problem. In connection with the first aspect Javier Gomez-Serrano described how physics informed neural networks can be used, after solving a minimisation problem, as initial guesses for approximate solutions to PDEs. This approach can potentially lead to fully computer assisted proofs of open problems regarding finite-time blow-up of solutions. The minimisation process is similar to the process in deep learning when neural networks are trained.

The second aspect of the discussion was devoted to DeepMind's program on computer assisted intuition in mathematics using deep learning. A critical part of this program is to use so-called saliency maps to determine the 'mathematical structure' that the neural network may have detected. The instability of such saliency map was brought up as an issue that mathematicians must be aware of. In particular, a tiny perturbation could potentially change the detected 'mathematical structure' substantially.







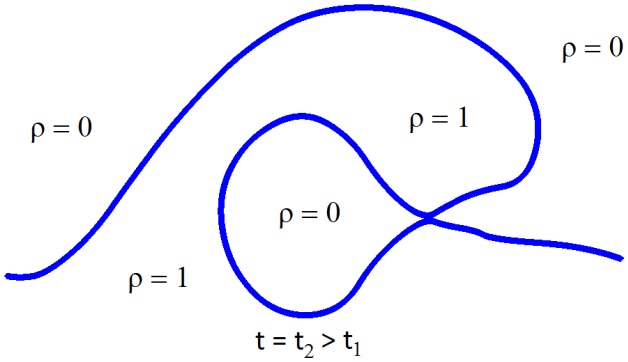


FIGURE 1. (**The water wave graph to splash**). From top to bottom: the figures W, X, Y, Z mentioned in Problem 6.