Snapshots of modern mathematics
from Oberwolfach

№ 6/2016

High performance computing on smartphones

Anthony T. Patera • Karsten Urban

Nowadays there is a strong demand to simulate even real-world engineering problems on small computing devices with very limited capacity, such as a smartphone. We explain, using a concrete example, how we can obtain a reduction in complexity – to enable such computations – using mathematical methods.

1 Certified computer simulations

Thanks to present-day computing power, it is possible to calculate the solutions to equations that model, to a varying degree of accuracy, real-world systems. Computing a set of solutions with input that represents different conditions of the system, we can then display it in succession (as images, for example) and get a *simulation* of the system. Computer-based simulations are omnipresent in today's life. Indeed, every day we see animations of weather systems, mechanical engines, and space probes fly-bys, to name a few examples. But what is the process that is used to produce the data that we see as pictures or movies? And how can we trust that these animations are "sufficiently close" to reality? Of course, using animations for teaching or entertainment purposes is nice and one might not usually wonder if such a simulation is (in whatever sense) correct. On the other hand, more and more important decisions are made based largely – or even solely – on computer simulations. Signing blueprints for a bridge based upon computer simulations of earthquakes requires that we can guarantee that the simulation is sufficiently accurate when compared with "reality" (such that

the difference is at most 5%, say). Writing this in a formula reads:

$$\textbf{distance}(\texttt{Reality}, \texttt{Simulation}) \leq \texttt{DesiredAccuracy}. \tag{1}$$

We call a simulation method *certified* if we can mathematically prove (1) for it. There are some questions and challenges that immediately arise from this desire:

- One first needs to detail `Reality`. The process of translating real phenomena into mathematical problems is called mathematical modeling. It might be said that this is what natural sciences are very much concerned with.[1] Depending on the stage of scientific development and purpose of the model, these models may vary greatly in the accuracy by which they resemble the real phenomena. As an example, take a model for a ball thrown in the air: On one end of the complexity spectrum we have a very simple model which translates to the (fairly) simple equation $h(t) = -4.9t^2 + v_0 t + h_0$. This model only takes into account earth's gravity ($g/2 \approx -4.9$), the ball's initial velocity (v_0), and its initial height (h_0). On the other end of the complexity spectrum we might have a model that also takes into account air resistance, side winds, humidity, the exact shape of the ball, and more. The second model is obviously much more accurate, but also much more complex. It has to be verified that the mathematical problem (model) represents the real process sufficiently well. The mathematical analysis is then devoted to the question whether the mathematical problem admits a unique solution that is robust with respect to changes of input data.[2]
- One needs to specify in which sense **distance** in (1) is to be understood. The simulation method to be chosen and the mathematical analysis needed to ensure (1) crucially depend on this choice.
- We do not know `Reality`. If we did know it, there would be no need for a simulation. Thus, also the distance on the left-hand side of (1) is unknown. Similarly, in many cases (for example, turbulent flow), it is not even known if a solution of the corresponding mathematical problem exists (even though, of course, we can sometimes observe the real process by experiments).
- It is straightforward to assume that the computer simulation is more "costly" for smaller tolerances. The finer the resolution of the simulation, the more computer power (in terms of memory, computing time, number of processors, ...) will be needed. Moreover, increasing the complexity of the problem to be simulated also requires increasing computing power.

[1] For an example detailing a process of modeling see Snapshot 7/2015 *Darcy's law and groundwater flow modelling* by Ben Schweizer.

[2] For an example of a system which is very sensitive to differences in data see Snapshot 5/2015 *Chaos and chaotic fluid mixing* by Tom Solomon.

Computers have become enormously powerful over the last decades. However, this fact does not mean that all relevant simulation problems can nowadays be easily solved simply by using supercomputers:

1) The complexity of relevant problems has increased much faster than the available computer power. For example, simulations of the human brain or of quantum physical networks are completely out of reach.
2) More and more machines are controlled by micro-chips that serve as local mini- or micro-computers in sites that demand simulations; for example, hospitals and industrial facilities. These local micro-controllers have very limited capacity both in terms of memory and in terms of computing power.

Real-time on-site computing can be achieved in two ways. In the "cloud" approach, the necessary computations are performed very rapidly – but on remote servers, and then the outcome is transferred to the local machine. Alternatively, in the "onboard" approach, all the necessary computations are performed directly on the micro-controller. The onboard approach obviates the need for any (slow, or fast) network connectivity and can thus be important in fail-safe situations; conversely, the cloud approach enlists much more significant resources, in particular storage, and hence permits treatment of larger problems. In this snapshot, and in our example, we emphasize the onboard approach, yet methods we describe here can be readily incorporated into either onboard or cloud embodiments.

2 Dynamic positioning of offshore supply vessels

The demand for producing (also alternative) energy has increased the need for offshore (oil drilling, wind turbine, wave energy) platforms. These platforms have to be supplied by offshore supply vessels; see Figure 1. Conditions out at sea are often rough: strong and changing winds, towering waves, dangerous currents. Offshore supply vessels must be able to operate in even the most adverse weather. And at the same time, they must maneuver with the utmost precision to ensure that platforms can be approached safely. The aim is that the supply vessel can keep its position even in wind and waves, while maintaining minimal drifting. The process of such a maneuvering is called *Dynamic Positioning (DP)*; see [4, 6, 8, 12] for more details.

The mathematical model for the DP gets weather and water conditions (in terms of measurements) as input and produces a control function to the ship's propulsion and steering systems in such a way that the vessel keeps its position relative to the platform as much as possible. In addition, the required energy for the propulsion has to be minimized. Obviously, this requires a mathematical model for the motion of the ship on the water surface, which is a

highly complex mathematical problem (see, for example, [11]), and can only be simulated by supercomputers and at considerable expense – at least several hours of computing time. On the supply vessel, however, DP has to be done in real time using a relatively small onboard controller, comparable to a small notebook.

Figure 1: Dynamic positioning for offshore supply vessels.

3 Computing on a mobile device

How to perform DP on the onboard controller? We will use "smartphone" as a synonym of such a small device with very limited capacity in terms of memory and computing power, plus the desire to do the computations in realtime.

What is the difference between a supercomputer and a smartphone? This seems to be a question with an obvious answer: The computing power of a supercomputer is by far larger than that of a smartphone – or any local micro controller. The DP onboard-unit is nothing more than a small computer similar to a smartphone. We would like to be able to do all DP computations on such a smartphone (with no connection to a "cloud"), keeping in mind that a fully detailed simulation would require several days on a supercomputer. Clearly, a smartphone is not a supercomputer, but perhaps a smartphone can produce results that achieve the same effect as those of a supercomputer. But how to achieve this?

Parametric problems. Putting the DP problem in the terms of inequality (1): `Reality` is the ship propulsion control and `Simulation` its High Performance Computing (HPC) simulation (that is, the simulations performed on a supercomputer). However, both depend on the particular value of the parameters, namely weather and water conditions (waves, currents), so we write `Reality`(p) and `Simulation`(p), where p encodes all above mentioned input parameters.[3]

[3] And so the equations describing the propulsion and simulation are called *parametric equations* and the problem to solve is a *parametric problem*.

The onboard controller (our "smartphone") should be able to approximate the dependency of the propulsion control on p fast (in realtime) and in a certified manner. That is, the onboard controller should provide a solution $\texttt{Smartphone}(p)$ for all reasonable parameters p, that fulfills

$$\textbf{distance}(\texttt{Reality}(p), \texttt{Smartphone}(p)) \leq \texttt{DesiredAccuracy}. \tag{2}$$

4 Model reduction and the Reduced Basis Method (RBM)

To achieve our goal we need a method for "reducing" a mathematical model so even a smartphone can produce solutions from this model. The Reduced Basis Method (RBM) has been proven to be a highly efficient model reduction technique. It allows for an onboard efficient reduced simulation for parameterized problems (those given in terms of parameters). We will now introduce the main ingredients of the RBM.

Separation of offline and online phase

The first idea consists of an appropriate combination of the use of both HPC and the onboard unit (the smartphone). Well before the supply vessel leaves the harbor to the offshore platform, we do have time for HPC — this phase is called *offline*. An RBM aims at finding an appropriate reduced model in the offline phase that is later invoked onboard to determine the actual propulsion control in the so-called *online* phase. In other words, we precompute an "Online Dataset" in the offline phase.

This is done as follows: Recall that the parameter p contains weather and water conditions, say (i) wind speed, (ii) wind direction, (iii) wave height, (iv) wave length, (v) wave phase and (vi) wave main direction. This means that p is a vector consisting of six entries. So the problem involves solving a complex equation describing complicated relations between six quantities varying over a large range. This might be possible for a supercomputer, but is way beyond the capabilities of a smartphone. We want to reduce this complexity. One can think of p as residing in a six-dimensional space ($p \in \mathbb{R}^6$). A way to reduce the complexity of the problem is to reduce the dimension of the problem by *projection*. This method can be pictured as projecting a shadow of a person on the floor and checking whether it is still possible to recognize the three-dimensional person by looking at their two-dimensional shadow. Recall that we wish to guarantee that the reduced system allows for the bound in (2), so we would like to reduce the system enough for its computations to be performed on the onboard unit, but not so much that it is no longer accurate enough. Usually, the maximal capacity of the onboard unit (in terms of memory and computing power) is known and can be translated into a maximal dimension of the reduced

system such that the Online Dataset can be stored and the computations can be performed in realtime. Let us call this maximal size $N_{\max}$ (say, 100). Bear in mind, though, that $N_{\max}$ itself might not be sufficient for the reduced system to be a certified one.

The reduced system is generated by reducing the system to a one-dimensional one and successively increasing the dimension – of course assuming (and mathematically proving) that increasing the dimension results in a better quality of the reduced system. For every dimension n ($1 \leq n \leq N_{\max}$), write $\texttt{Reduced}(n, p)$ for the solution of the reduced system of dimension n for a specific parameter p. Then, ranging over all possible p's, find the parameter for which **distance**$(\texttt{Reality}(p), \texttt{Reduced}(n, p))$ is maximal. Denote this parameter by $p^{(n)}$. If this distance is smaller than (or equal to) `DesiredAccuracy`, we have found a certified reduction – $\texttt{Smartphone}(p)$. We know this since we chose $p^{(n)}$ to be the point that gives maximal distance, so for all other values of p the distance is even smaller and certainly adheres to the restriction in (2). If that is the case for all p then (1) is also satisfied and we have a certified simulation. If the distance for $p^{(n)}$ is larger than `DesiredAccuracy`, we increase the dimension n and try again until we reach a dimension N for which (2) holds or stop when we reach $N = N_{\max}$. The computation of $\texttt{Reality}(p^{(n)})$ does not go to waste, though, as we use it to determine the basis of the reduced system – in the wording of the shadow analogy, we use it to position the lights in a way that achieves the most recognizable shadow.

Let us summarize this process in the form of an algorithm:[4]

Algorithm 1 Greedy algorithm.

1: **for** $n = 1, \ldots, N_{\max}$ **do**
2: choose $p^{(n)}$ as the maximal value of
distance$(\texttt{Reality}(p), \texttt{Reduced}(n, p))$;
3: **if** $\{$**distance**$(\texttt{Reality}(p^{(n)}), \texttt{Reduced}(n, p^{(n)})) \leq \texttt{DesiredAccuracy}\}$
$N = n$; **STOP**;
4: update reduced model by $\texttt{Reality}(p^{(n)})$;
5: **end for**.
6: $N = N_{\max}$; % If `DesiredAccuracy` is not reached.

[4] An *algorithm* is a finite sequence of instructions to be performed step-by-step. Algorithms are used most widely to describe the structure of computer programs. Note, not all algorithms describe a finite process; for example: "**1:** Move one step forward. **2:** Repeat **1**." This was an example of a (infinite) *loop* which is one of the most common structures in algorithms. Another form of a loops is "**for** each value in a range **do** some operations". Another common structure is the *conditional*: "**if** some condition holds **then** perform some operation".

When Algorithm 1 terminates, we either obtain a system of dimension $N_{\max}$ (which still might not satisfy (2)) or a system of smaller dimension $N \leq N_{\max}$ which guarantees the accuracy demands (2). A couple of comments to the above algorithm are in order:

- This method is called "greedy"[5] since we take the maximal value in line 2;
- The question if such a scheme terminates and, if yes, how fast, has been a field of active mathematical research over the past years;
- The true distance in line 2 is not computable, since `Reality` is unknown. It is part of current mathematical research to construct a possibly tight computable upper bound for this unknown distance;
- The maximal value in line 2 is usually taken over a test subspace of parameters, whose construction might be delicate;
- In (2), we seek some $\texttt{Smartphone}(p)$ with distance `DesiredAccuracy` from $\texttt{Reality}(p)$. If we use $\texttt{Reduced}(N, p)$ with the terminal value $N < N_{\max}$ as $\texttt{Smartphone}(p)$, then Algorithm 1 in fact ensures the error bound (2);
- For the computation of $\texttt{Reality}(p^{(n)})$ (which is called *snapshot* in RBM) in line 4 we use the HPC-based supercomputer output $\texttt{Simulation}(p)$.

Why should this work?

After the precomputation of the reduced model in the offline phase using the above greedy method, one can also precompute several ingredients for the reduced system, so that this system (the Online Database) can then completely be stored on the micro-controller. Then, given a new value for the parameter (for example, current measurements for weather and water), the reduced system can be solved highly efficiently on the small device.

Of course, one could construct such a reduced system also by other techniques, for example by simplifying physical relevant phenomena or neglecting certain effects.[6] This, however, would not necessarily yield a *certified* reduced simulation. One might also think to simply use the precomputed snapshots as a database and to use the "closest" parameter constellation in the online phase, however this is often very inefficient (for reasons of both accuracy and storage).

The question remains, why or under which circumstances this works at all? One main observation is shown in Figure 2, where the error (the distance between the true and the reduced solution) is depicted over the dimension n of the reduced

[5] A *greedy* algorithm is one that breaks the problem to be solved into steps (passing over them via a loop) and chooses the best solution at each step as if that step was the problem to solve. There are many cases in which a greedy algorithm does not come up with the best solution to the full problem or with a solution at all. However, in our case it is a suitable strategy.

[6] As in the thrown-ball model above, where the very simple model did not take into account air resistance and other aspects influencing the ball's trajectory.

system (here for the so-called allocation, a major part of the full DP problem). It turns out that for many different real problems, there is a very fast error decay, so that a reduced system of very small dimension suffices to realize the desired accuracy for different parameters. For some of such problems, there is a rigorous mathematical proof of such a fast decay, for others there is at least experimental evidence by simulations. Mathematicians say that the "solution depends smoothly on the parameter". It is an open problem of current research to quantify what kind of smoothness yields which kind of error decay.

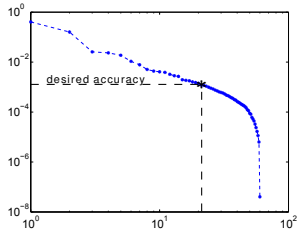


Figure 2: Error decay (accuracy 10^{-3} reached here for $n = 112$).

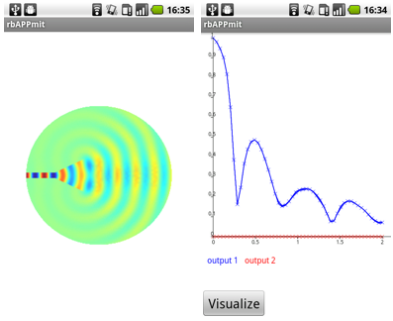
5 RBMs on smartphones

We present here some results of the first smartphone implementation of the Reduced Basis Method, presented in [7]. The code is implemented in the programming language Java and the online phase is executed on a Nexus One Android-based Smartphone with a 1GHz processor and 512 MB of memory with double-precision accuracy. The OpenGL ES Library is invoked for graphical renderings. Several different parametrized problems (here, parametrized partial differential equations (PDEs))[7] are loaded on the smartphone, including linear two- and three-dimensional problems, and nonlinear problems. Note that memory restrictions limit both the number of problems we can consider and the size of each problem.

We present here results of the reduction method on the smartphone described above for one particular problem: an acoustic horn, modeled by the (frequency-domain) Helmholtz PDE. The parameters here are the geometry of the flare and the frequency of the excitation (incoming pressure wave). For this problem, N is quite small, $N = 53$, and hence online response time is very fast: roughly one or two seconds from specification of the parameters to completion of the smartphone solution; recall that all calculations are performed onboard. In Figure 3(a), we show the spatial dependence of the pressure field for particular parameter values; in Figure 3(b), we show a plot of the modulus

[7] *Partial differential equations* are equations that describe a relation between a function of several variables and its partial derivatives with respect to these vriables. These can often be extremely hard to solve and, sometimes, a solution does not exist at all.

of the reflection coefficient of the horn as a function of the frequency. We emphasize that both panels are direct "screen grabs" from the Nexus Smartphone. Figure 3(b) shows $\texttt{Smartphone}(p)$ together with rigorous lower and upper bounds for $\texttt{Simulation}(p)$ based on *a posteriori* error estimates[8] for $\textbf{distance}(\texttt{Simulation}(p),\texttt{Smartphone}(p))$. In fact, $\texttt{Smartphone}(p)$ (indicated by the $\times$'s) is sufficiently accurate that the lower and upper bounds for $\texttt{Simulation}(p)$ (indicated by two solid lines) essentially collapse to a single curve, more or less through the $\texttt{Smartphone}(p)$ "$\times$" marks.



(a) **(b)**

Figure 3: Smartphone simulation for an acoustic horn.

6 Conclusions and outlook

Our title, "High Performance Computing on Smartphones", is justified because – by virtue of certification – the results from the smartphone are in a sense equivalent to the results we would have obtained had we directly invoked $\texttt{Simulation}(p)$ on a supercomputer. But of course, $\texttt{Smartphone}(p)$ is much faster, and more suitable for on-site, real-time, fail-safe applications.

The Reduced Basis Method was conceived many decades ago and has enjoyed a renaissance in research and development in the last fifteen years; see [1, 5, 9, 10], to mention just a few. Many researchers across the world have made important contributions, as demonstrated by the workshop series www.morepas.org and recent research articles in [2]. We hope in the future to fully address problems as ambitious as the Dynamic Positioning grand challenge described in Section 2.

[8] These are error estimates derived from experiments using statistical methods.

Image credits

Figure 1 Kindly provided by D. Jürgens, Voith Turbo Schneider Propulsion, Germany.

Figure 2 From [3], by Karsten Urban and Anke Brandner.

Figure 3 From [7, Figure 1].

References

[1] B. Almroth, P. Stern, and F. Brogan, *Automatic choice of global shape functions in structural analysis*, AIAA Journal **16** (1978), 525–528.

[2] P. Benner, M. Ohlberger, A. T. Patera, G. Rozza, D. Sorensen, and K. Urban (edsitors), *Model Reduction of Parametrized Systems (MoRePaS)*, Special Issue of Advances in Computational Mathematics **41** (2015).

[3] A. Brandner, *Optimale Allokation für Voith-Schneider Propeller im Rahmen des dynamischen Positionierens*, PhD thesis, Ulm University, 2015.

[4] C. Canuto, T. Tonn, and K. Urban, *A posteriori error analysis of the reduced basis method for nonaffine parametrized nonlinear PDEs*, SIAM Journal on Numerical Analysis **47** (2009), no. 3, 2001–2022.

[5] J. P . Fink and W. C. Rheinboldt, *On the error behavior of the reduced basis technique for nonlinear finite element approximations*, Zeitschrift für Angewandte Mathematik und Mechanik **63** (1983), 21–28.

[6] M. Hopfensitz, J. C. Matutat, and K. Urban, *Numerical modelling and simulation of ship hull geometries*, Progress in Industrial Mathematics at ECMI 2010, Mathematics in Industry, vol. 17, Springer, 2012, pp. 465–472.

[7] D. B. P. Huynh, D. J. Knezevic, J. W. Peterson, and A. T. Patera, *High-fidelity real-time simulation on deployed platforms*, Computers & Fluids. An International Journal **43** (2011), 74–81.

[8] D. Jürgens, M. Palm, S. Singer, and K. Urban, *Numerical optimization of the Voith–Schneider Propeller*, Zeitschrift für Angewandte Mathematik und Mechanik **87** (2007), 698–710.

[9] A. K. Noor and J. M. Peters, *Reduced Basis Technique for Nonlinear Analysis of Structures*, AIAA Journal **18** (1981), 455–462.

[10] G. Rozza, D. B. P. Huynh, and A. T. Patera, *Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: Application to transport and continuum mechanics*, Archives of Computational Methods in Engineering **15** (2008), 229–275.

[11] J. Springer and K. Urban, *Adjoint-based optimization for rigid body motion in multiphase Navier–Stokes flow*, SIAM Journal on Scientific Computing **37** (2015), B185–B214.

[12] T. Tonn, K. Urban, and S. Volkwein, *Optimal control of parameter-dependent convection-diffusion problems around rigid bodies*, SIAM Journal of Scientific Computing **32** (2010), 1237–1260.

Anthony T. Patera *is the Ford Professor of Engineering at the Massachusetts Institute of Technology (MIT, USA).*

Karsten Urban *is Professor for Numerical Mathematics at Ulm University (Germany).*

Mathematical subjects
Numerics and Scientific Computing

Connections to other fields
Engineering and Technology, Physics

DOI
10.14760/SNAP-2016-006-EN

Snapshots of modern mathematics from Oberwolfach are written by participants in the scientific program of the Mathematisches Forschungsinstitut Oberwolfach (MFO). The snapshot project is designed to promote the understanding and appreciation of modern mathematics and mathematical research in the general public worldwide. It started as part of the project "Oberwolfach meets IMAGINARY" in 2013 with a grant by the Klaus Tschira Foundation. The project has also been supported by the Oberwolfach Foundation and the MFO. All snapshots can be found on www.imaginary.org/snapshots and on www.mfo.de/snapshots.

Junior Editor
Daniel Kronberg
junior-editors@mfo.de

Senior Editor
Carla Cederbaum
senior-editor@mfo.de

Mathematisches Forschungsinstitut
Oberwolfach gGmbH
Schwarzwaldstr. 9–11
77709 Oberwolfach
Germany

Director
Gerhard Huisken