Quantum Entrepreneurship Lab: Final Deliverable

This report elaborates on the development of our start-up ***QDynamics***. From a business perspective, this report presents an analysis of the current market and the likelihood of success in the current economic environment. We make a case for the latter by emphasizing the needs of our potential customers and our corresponding value proposition. Moreover, a go-to-market strategy is outlined and correlated with its risk assessment. Finally, we delineate a pricing scheme and the projected milestones for our progress.

The second half of the report details the technical aspects of our product. We provide the theoretical background on partial differential equations and their uses in computational fluid dynamics. Additionally, two approaches to solve the problem are elaborated on, followed by an overview of our solution and implementation. Lastly, a potential system architecture for our solution is described and put in context together with a timeline for the implementation of our solution.

QDynamics

Table of Contents

QDynamics

Our Business Model

To be successful as a Start-Up we also had a in-depth look at the business model.

Need Analysis
From our in-depth market research and interviews we conducted, with engineers and industry experts, we identified several problems in the CFD process. Current solutions cannot handle simulations of more complex parts. In practice, this means that it is not possible, for example, to completely simulate one entire aeroplane wing. This leads to the necessity of splitting the model up into smaller parts and simulating them one at a time, which is a tedious and long process for the engineers. Even the best computers in the world cannot solve this problem of really long idle times between iterations.

Engineers only have one option to solve this problem, which is to test their design in facilities like wind tunnels, which can costs thousands of dollars per hour. In summary, this results in a suboptimal experience for the engineer, extremely long development cycles for new designs, and huge costs for the aerospace companies.

Value Proposition
To solve the problems we identified during our research, we are proposing a software solution that will help engineers in the aerospace industry to perform accurate and rapid prototyping of their designs.

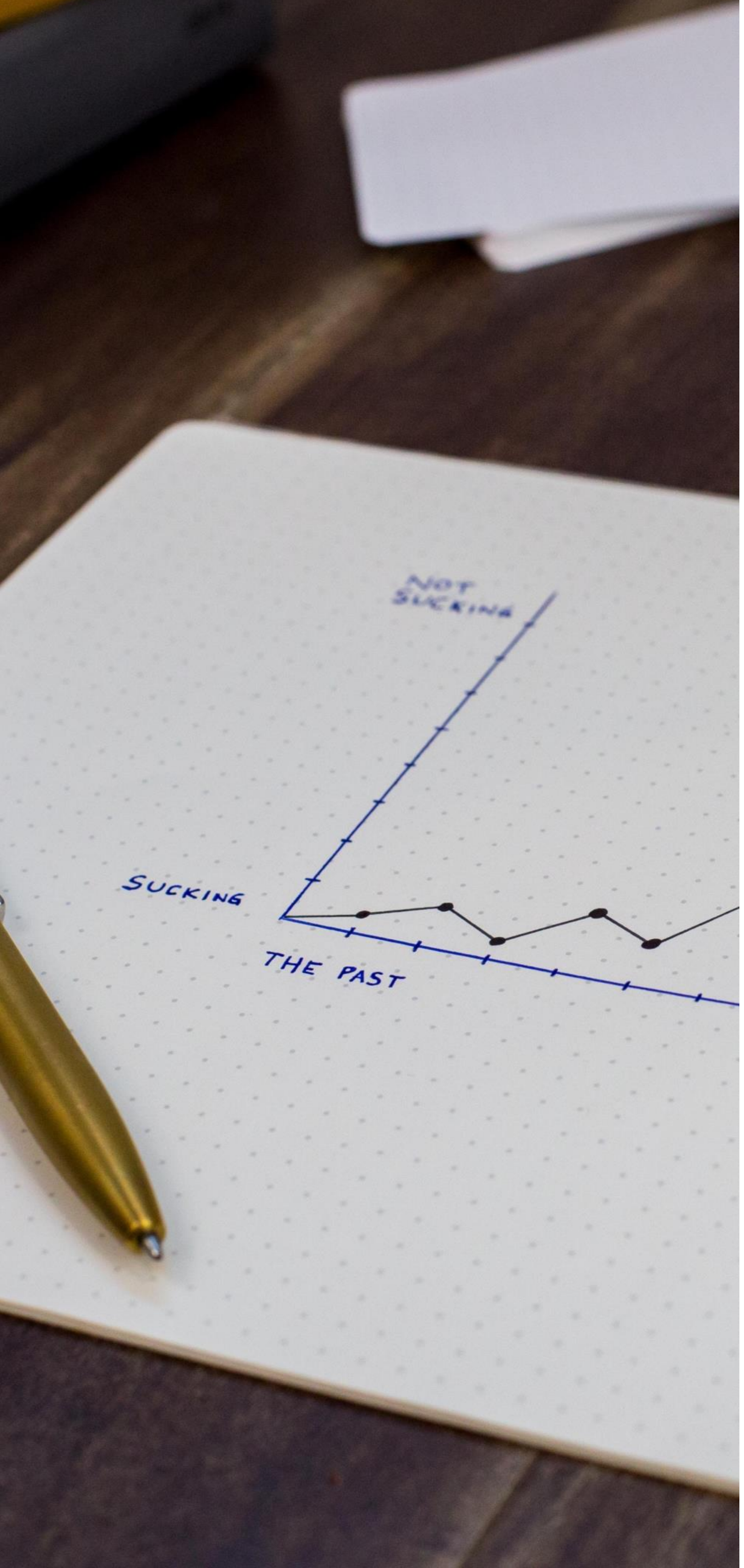
With this solution, we are aiming to increase the precision compared to current CFD prototyping solutions while significantly reducing the runtimes of these simulations. This will result in a much more seamless process for the engineers who will be able to quickly cycle through and test their designs. As a result, faster simulations will yield more iterations, leading to a more structurally sound and efficient design process. Moreover, less time spent on simulations means more time spent on improving the actual design of the parts.

The top 3 benefits of our solution are:

1*) Reduced time to market*
Our solution and API will significantly improve the time to market for new airplane components, as more parts can be tested in the same amount of time, while maintaining (and possibly increasing) the level of accuracy.

2*) Significant cost savings*
Less reliance on physical testing facilities like wind tunnels decreases the total cost spent on a new component by a big margin. Secondly, faster iteration times mean that the workforce can be utilized more efficiently.

3) *More user-friendly simulations*
One of the biggest advantages of our solution is, that the design process is more seamless for the engineer. We achieve that by reducing runtimes and by making our solution as easy to integrate as possible. Our users will be able to install and use our solution without having any knowledge about quantum technology. This is in line with one of our most important values: *„Bringing quantum to the people".*



Target Group

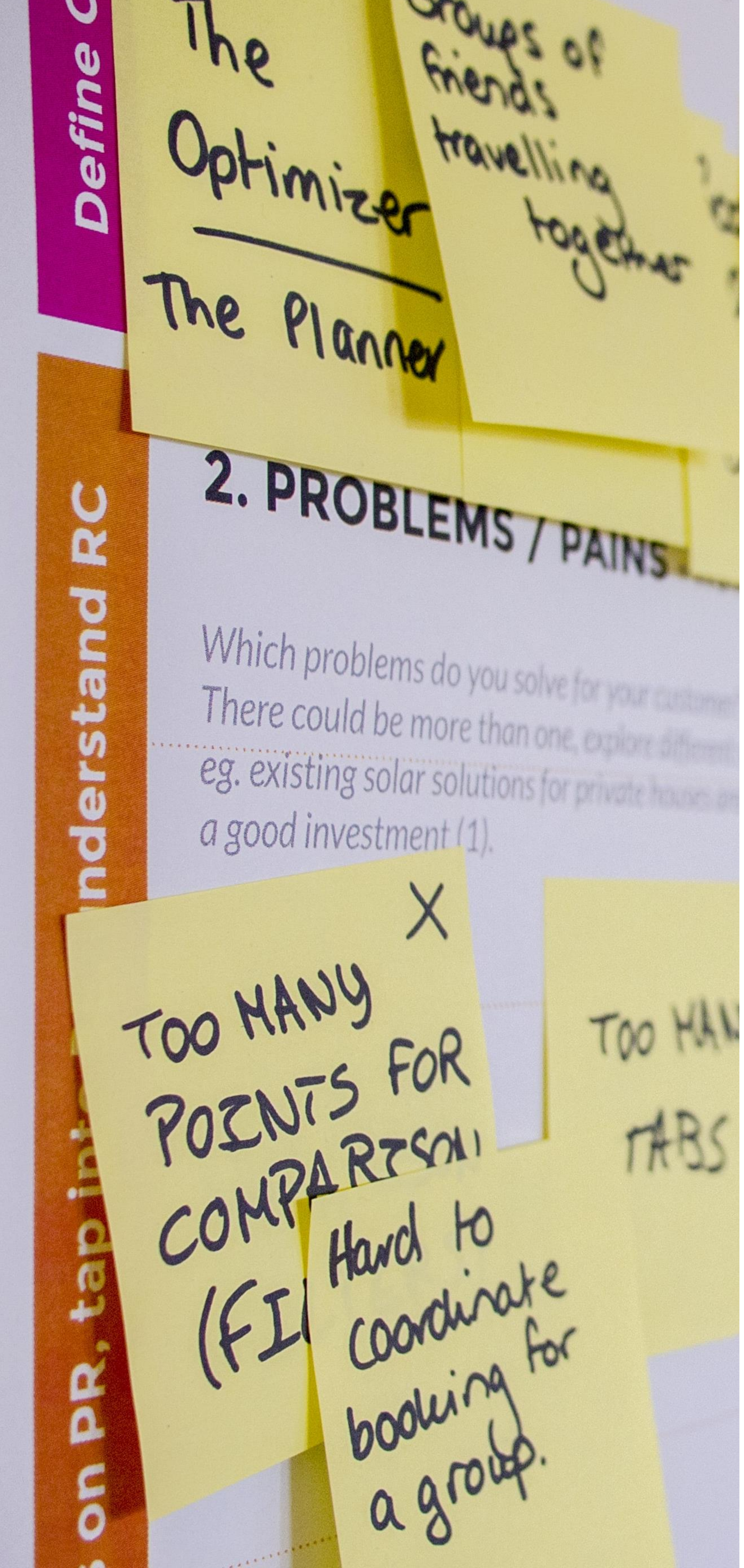
Who are we targeting with our product? To answer this, it is important to make the distinction between the users and the customers. Since we are trying to sell our solutions to big companies, our customers will rarely be the users at the same time.

When talking to Airbus and other players in the industry, we found out that most project managers of teams are responsible for sourcing software solutions that their team is working with. For more broadly used software in the company, the chief technology officer often makes the decision if a new solution should be bought and implemented. Especially with big companies like Airbus, we identified the necessity of custom solutions, since there are often very specific requirements and processes that the software has to be integrated into. Therefore, we chose to start with an API, so we can best address this need of our target customers. The goal of our target customers is to enable their team to work more efficiently and consequently provide better products. On a higher level, they also want to be seen as an industry leader and stay ahead of their competition.

So now that we know who our target customer is, we are going to have a look at the people that will use our product: our target users. They usually work within the teams of our customers as aerospace engineers. Their daily work mostly consists of designing new parts, testing them, and meeting regulatory standards for the safety of their designs. The problem they have is that they have more design ideas than they can test within a reasonable amount of time, because of the long iteration times caused by current simulators. Also, they typically have very little knowledge about the inner workings of CFD software and quantum technology. We, therefore, ensure that integration with our solution is as easy as possible.

Go-to-Market Strategy

As a Start-Up we want to grow with quantum since the current technological progress is not quite at a point where it can solve all the difficulties of CFD. Therefore, we intend to be at the forefront of innovation in the quantum CFD field and help further its progress. To enter the market, we are planning to start with an API to our PDE solver which will be easy to integrate into open-source software. It can also be used in custom solutions which are common with the big players in the aerospace industry. This API is the appropriate first product because it allows us to get to know the market and its needs even better and we can continuously improve on this infrastructure. The choice to start with an API is also in line with the research and interviews we have done within the industry. We found out that, especially in the aerospace industry, there is a necessity for custom solutions, as each company has very specific and niche requirements. This fact also requires us to work together closely with our customers and provide customization services as an add-on with a consulting and implementation fee. The underlying algorithm is already under development and we will be able to launch the first iteration of the solution sometime in 2022. At a later stage, we will also expand our offering with a complete multi-physics simulation suite. To protect our intellectual property, we will quickly move to try secure the necessary patents and exclusive collaborations with clients. In comparison to the already existing players in the CFD space, our advantage mainly consists of speed to market.


Define
The Optimizer
The Planner
Groups of friends travelling together
2. PROBLEMS / PAINS
Which problems do you solve for your customer?
There could be more than one, explore different
eg. existing solar solutions for private houses are
a good investment (1).
Understand RC
on PR, tap into
TOO MANY POINTS FOR COMPARISON
TOO MANY TABS
Hard to coordinate booking for a group.

The biggest CFD players usually take up to 10 years to adapt to new standards and emerging technologies. Therefore, this early entry advantage will greatly improve our success when entering the market.

Risks and Showstoppers

Start-Up projects especially in the deep tech space have a high risk of failure and many risks that have to be considered and prepared for, before entering the market with a product. Therefore, we thoroughly examined the risks we might run into. For instance, we found an existing patent by Qu & Co [1], which could have an overlap with the approach we came up with. We are in contact with the patent owner and are still investigating the overlap and are aware that a big overlap could result in severe licensing fees or could even prevent us from using this approach in our products.

Another threat is that the CFD market mostly consists of big players and very few smaller startups. Therefore, the market is difficult to successfully enter and a proper go-to-market strategy is of paramount importance. We strongly believe that our approach of starting with an API and closely collaborating with big clients will enable us to quickly be known in the market. Direct competition in the quantum space currently is very sparse with BosonQ Psi from India, being the closest competitor as they are also leveraging quantum computing. To differentiate from them, we are starting in the very narrow niche of aerospace CFD, while the other start-ups have a much broader approach and are targeting the whole Multiphysics simulation market.

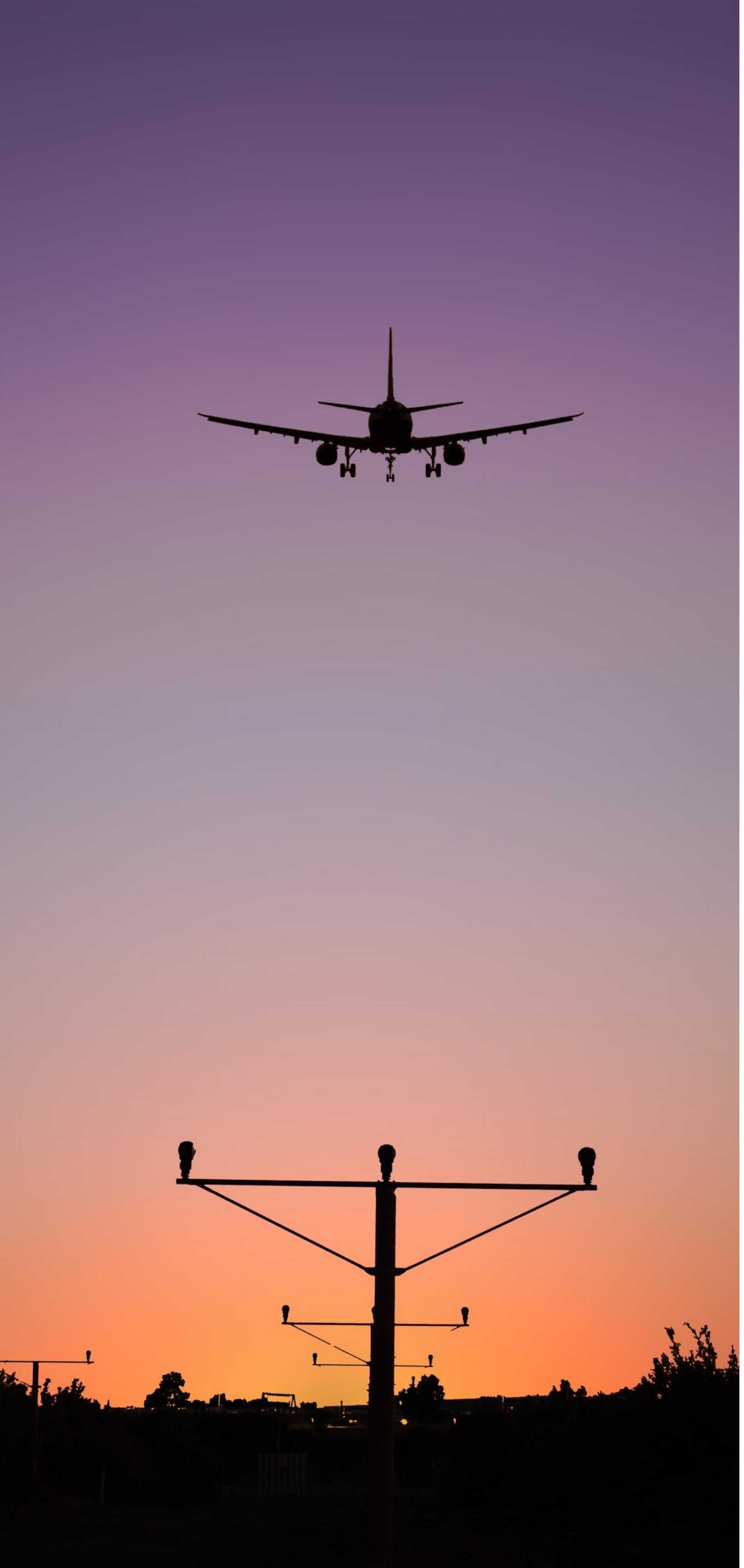
Our Team

The biggest reason why our idea is going to succeed is our amazing interdisciplinary team. We are made up of a unique and diverse team of students and working professionals from a range of different fields. Our expertise in the technical domain as well as in the business space will ensure that all necessary areas are cared for.

Market Potential

While our idea is very interesting and a promising innovation for a lot of industries, we also evaluated the market potential for our solution. During our research, we started with a top-down approach and the estimations of the CFD market from various research institutes. Worldwide, the entire CFD market amounted to around USD 4.5 billion, according to firms like Technavio [2]. This represents our *Total Available Market*. In this market, we calculated a *Serviceable Available Market* of USD 600 million, based on cost savings as we are eliminating the need for expensive physical testing and making the entire design process faster. This number aligns with the estimations of the Boston Consulting Group from 2019. Lastly, we calculated our *Serviceable Obtainable market* based on our revenue projections within 5 years. This amounts to USD 26 million assuming that we only target the EU aerospace market, and we can capture a 4% market share in this space. This number may sound ambitious within 5 years but is achievable when we target working with bigger clients such as Airbus, which make up most of the market in the EU. We decided to target the EU market, and not specifically Germany. Germany alone is not as attractive as the EU and we would not be able to build on existing relations with European companies like Airbus.

This market already looks extremely promising, but it's growing even bigger. In 2020, the CFD market grew with an annual compound growth rate of 9% and the advent of fault-tolerant quantum computing will increase this number by a lot. BCG expects value creation in the CFD industry to reach USD 19 billion to USD 37 billion [3] when we finally achieve fault tolerance.



It is therefore a very good idea to enter this market early and focus on advances in the quantum space, to gain a reputation in the industry and position ourselves well for the future.

Our revenue growth projection after 2022 when we would launch the API solution is based on multiple assumptions, we made that lead to the mentioned 26 million USD.

We are assuming an average utilization of our solution per engineer or license of about 4 hours for 220 days per year. This number is in line or even underestimated with industry benchmarks. Moreover, we believe that we can acquire 2.500 users by 2026 and calculate with our pay-as-you-go pricing scheme.

Milestones

As already mentioned before, we are entering the market with an API that will rely on a hybrid solution of classical computation and quantum technologies by 2022. During this phase, we are going with a pay-per-use pricing model and want to build on existing relationships with Airbus and develop our solution together with them.

After our launch, we want to continue to grow with quantum and develop a full software solution by mid-2024. In the following years, we plan to quickly expand into other industries like the automotive industry or the energy sector as well as other geographical markets. As soon as the technology is ready, we want to switch to a full quantum-powered CFD speedup and provide an even better service to our customers.

Pricing

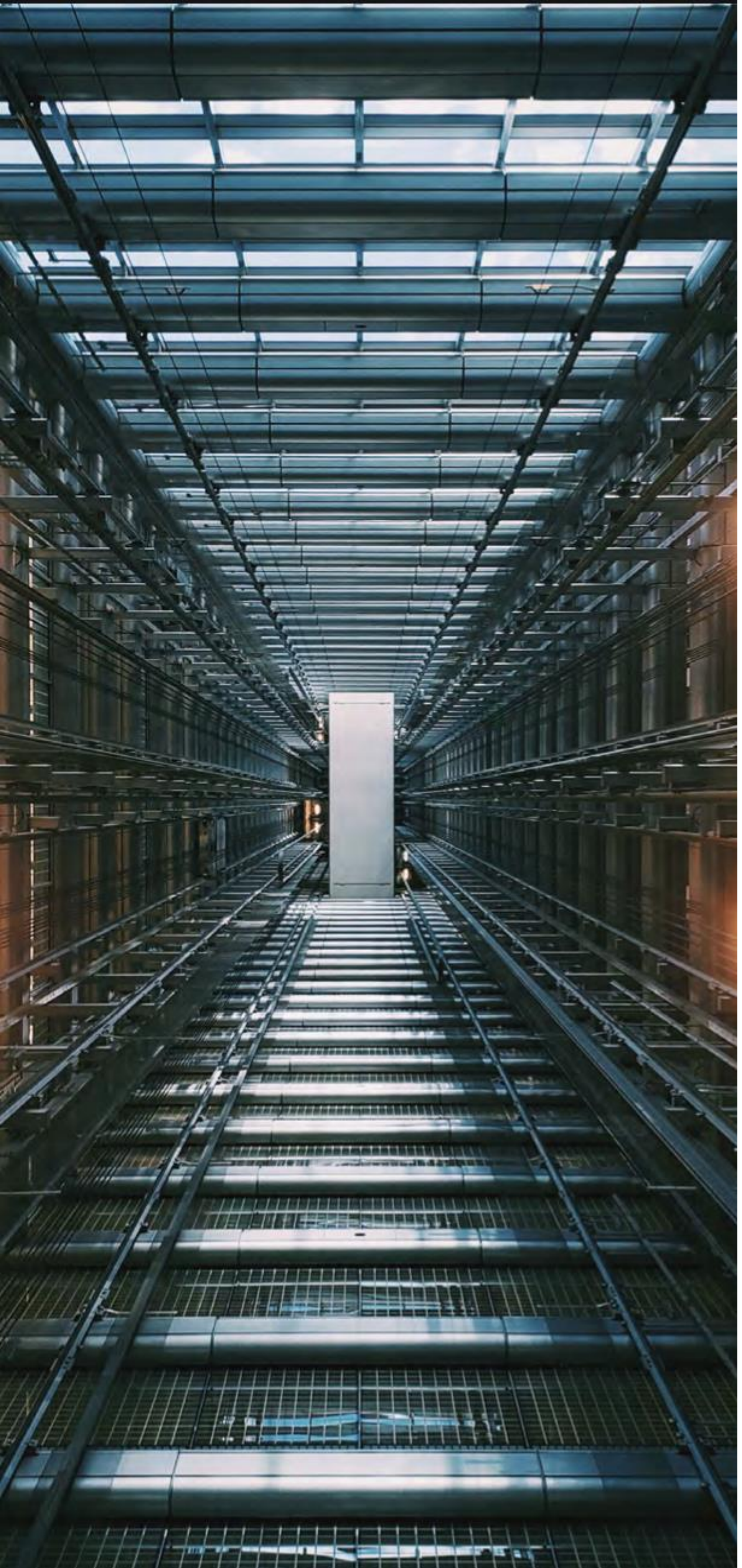
Initially, we are starting with the pay-as-you-go due to the fact that we need to be flexible in our offering. Moreover, this pricing scheme also enables prospective customers to more easily try out our solution with minimal involvement. It is also noteworthy that this is a common pricing scheme in our targeted industry. Upon expanding with different functionalities, we are planning to change to a subscription-based cost scheme, leading to more predictable cash flows for our company.

With pay-as-you-go, users pay for the minutes they are using our solutions. This can encourage more users to try out our solutions without commitment. In the initial release phase, the starting prices are $1.5 per minute for the simulator and 76$ for the quantum computing backend calculation, which is 95% of the total cost. These prices are based on a value-based estimation approach, calculating against the cost savings with our API compared to current solutions. This calculation also takes into account our costs including development, infrastructure, and support. The costs for the infrastructure can be seen in the table below. We believe that proper support for our clients is the main driver for the sustainability of our solutions, where it can be the seed that allows us to expand our company. Therefore, we want to significantly invest in close relationships with our customers.

About 2 years after launch, our projected margin in the steady-state will be around 70-80%, which is comparable to other SaaS companies.

	Infrastructure	**Estimated Costs**	**Price**
QC simulator	Amazon bracket, SV1 simulator	$0.075 per minute	$1.5 per min
QC calculation	ex. 1 task can take 10,000 shots total = task charges + shot charges total = $3.8 for an example calculation	$0.3 per task $3.5 per 10,000 shot	$76 per 10k shots calculation

Table 1: Estimated cost and pricing for the infrastructure



Technical Aspects

This documentation details the technical background of our product. Firstly, it serves to offer a brief overview of the formal background needed for solving the *Computational Fluid Dynamics* (CFD) and *Partial Differential Equations* (PDE) problems. Secondly, it touches upon the current classical solutions available for the aforementioned problems. Thirdly, it presents our solution by means of quantum hybrid neural networks. Afterwards, an analysis on the integration of our approach and its respective challenges is discussed. The proposed system architecture and the user journey is then described. Lastly, we illustrate the technical roadmap that we have envisioned from the ideation phase up to the eventual release of the product.

Solving Partial Differential Equations in Computational Fluid Dynamics

Solving Partial Differential Equations (PDEs) is a major challenge when tackling aerodynamic problems. A quantum (or hybrid) computing algorithm could be used for faster problem solving, potentially improving the costs traditionally associated to complex numerical schemes.

Problem Description

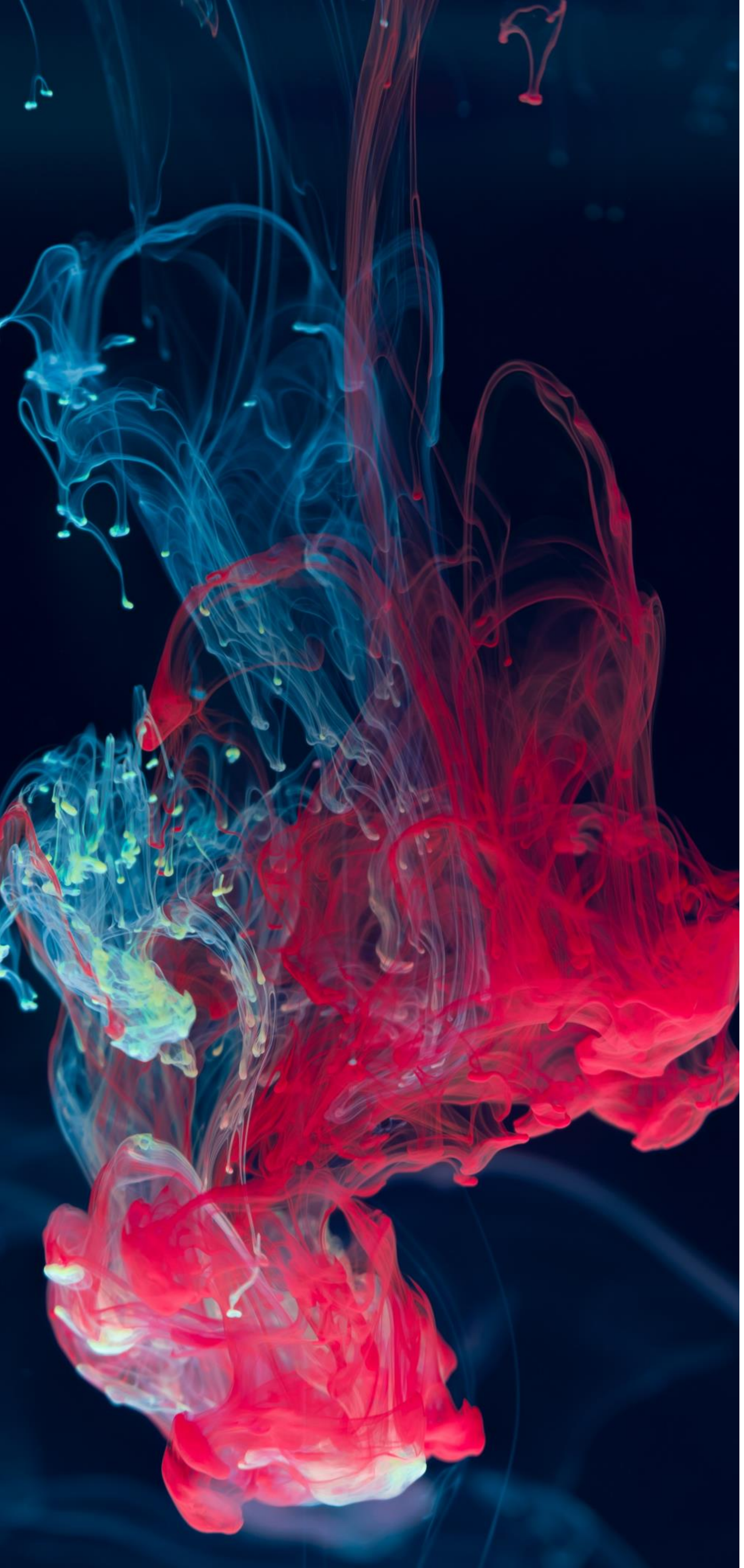
Solving PDEs is a crucial part in representing a wide variety of physical phenomena such as aerodynamics flow. Currently, PDEs can be numerically solved by approaches such as the Finite Difference, Volume, or Element Methods (FDM, FVM, FEM). They all rely on discretizing the space domain in which the equation is solved by means of a mesh or grid. Current simulation algorithms construct a fine, detailed mesh only in predetermined areas of interest and thus lose many possibly useful details outside these areas of interest. The associated computational cost grows as finer meshes and smaller time steps are required. Therefore, tasks in fields such as Computational Fluid Dynamics (CFD) are very time and resource consuming even for current state of the art High Performance Computing clusters. Furthermore, approaches leveraging on Neural Networks (NNs) have been published [4]. In their paper, *Raissi et al.* propose a new type of NNs, the Physically Informed Neural Networks (PINNs).

Solutions

We identified two different solutions to the problem:

1) Quantum hybrid methods based on Finite Element Methods, yielding numerical approximations of boundary value problems for partial differential equations, based on discretizing the parameter space via a finite mesh. Going for a quantum approach can result in polynomial speedups over the best classical algorithms for FEM [5].

2) Solving PDEs using modern deep learning techniques, building an approximate representation of the solution by means of a NN, the approach we opted for [6; 7]. The problem is formulated as a regression problem, solvable using classical machine learning algorithms. This removes the need of using a mesh as only randomly sampled points in the time and space domain are required. Moreover, the spatial and time dimensions are treated simultaneously. Also, it was shown that NNs can overcome the “curse of dimensionality” in solving (multi-dimensional) PDEs. However, in order to build a sufficiently accurate representation of the PDE, large neural networks and training times are required [8]. We demonstrate here how quantum algorithms can be applied to the generation and training of such neural networks based on current research in the area [9]. We assess the quality of the developed quantum algorithm with Burger's equation, which describes a particular case of aerodynamic flow.



Classical Approach

Our vision is to leverage quantum technologies in order to make CFD simulations faster and more accurate.

Nowadays CFD simulations rely on large FEM or FDM simulations and thus suffer from the so-called "curse of dimensionality". This states that the solution gets exponentially more difficult to compute with every dimension added to the initial problem. Research has proven, that NNs can overcome this severe limitation; the main reason we decided to investigate further on this approach [10].
An important building block to solving CFDs is represented by being able to efficiently solve partial differential equations (PDEs). To solve these PDEs, we use a particular type of feed-forward network, the Physics Informed Neural Networks (PINNs). Classical PINNs can best be explained with one of the most basic PDE in CFD: The (viscous) Burger's equation.

$$u_t + uu_x - \mu u_{xx} = 0,$$
$$x \in [-1,1], \qquad t \in [0,1], \qquad \mu = 0.001/\pi$$

We also define initial conditions (IC)
$$u(t=0,x) + \sin(\pi x) = 0,$$
and boundary conditions (BC_1 & BC_2)
$$u(t,x=-1) = u(t,x=1) = 0,$$
We then build a classical NN and use a method proposed by the authors of the original paper [4]:
We specify our problem as the loss function that the network trains on. Thus, the network is informed of the underlying physics needed to learn Burger's equation.
The loss function now resembles a least-squares problem, including the PDE, the IC, and BC1 and BC2.

$$loss = PDE^2 + IC^2 + BC_1^2 + BC_2^2 = [u_t + uu_x - \mu u_{xx}]^2 + [u(0,x) + \sin(\pi x)]^2 + [u(t,-1)]^2 + [u(t,1)]^2.$$

The derivatives of $u(t,x)$ in x and t are given by the gradient in the respective direction, which need to be calculated by the network for the gradient descent.
Raissi et al. proved that this also works for other PDEs like Schrödinger's equation very well, yielding a general approach on solving PDEs using NNs [4].

Figure 1 shows our solution $u(t,x)$ for Burger's equation in x and t and also $u(t=t^*,x)$ in x at several fixed t^* in the three lower plots. We found this solution with a feed-forward network consisting of eight densely connected hidden layers of width 20. The accuracy of our classical solution is above 96% and trains in roughly 10 minutes. However, this is only one step of the overall CFD problem. We thus need to further minimize the computation time and achieve an even greater accuracy, as this could be included in crucial calculations for aircraft safety. These two factors are a major bottleneck in efficient, reliable CFD simulation and the main reason for exploring how quantum can benefit this approach.

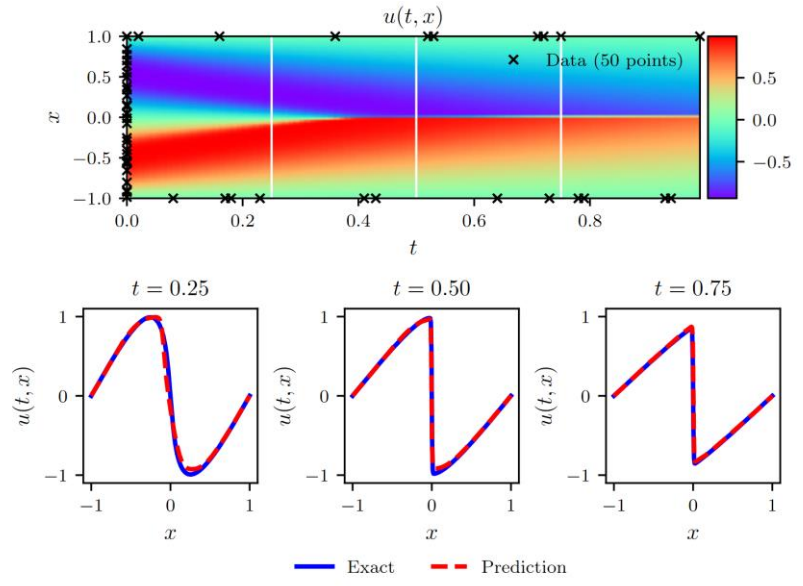
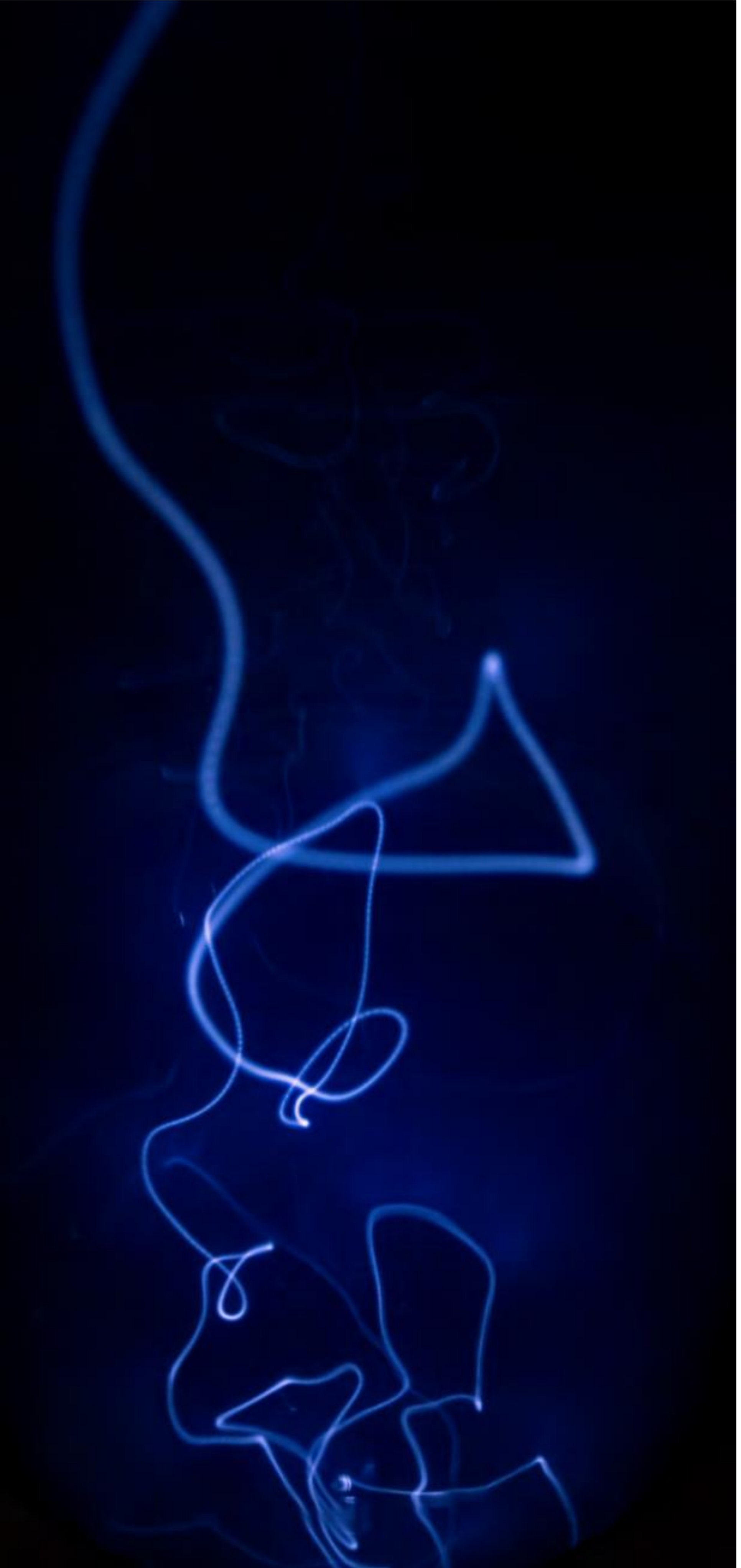


Figure 1: Solution to Burgers' equation. The PINN gives the solution $u(t,x)$ shown in the upper plot. In the three lower plots, we see that the predicted function is quite close to the actual solution. At $t^* = 0.75$, we can observe the formation of a shock.



Quantum Neural Networks

Speeding up neural network execution

While the usage of PINNs has the potential to greatly reduce the computational cost of solving PDEs, the accuracy of the results still depends on the size of the network [4]. Thus, using larger networks, in terms of layer size as well as the number of layers, can help to further increase the accuracy of the PINN. Moreover, the accuracy of the network can be increased by "calculating a running average of the neural network solutions over a sequence of training iterations" [6]. However, both these measures would further increase the computational cost of the network's training and execution.

This is where quantum computing enters the picture. Several studies have successfully achieved equally accurate solutions within a shorter time frame by using various implementations of *quantum neurons* in their neural networks [11; 12; 13]. This was done by measuring the results of their respective learning tasks while not increasing the network size. Thus, we intend to improve our PINN's performance by replacing some of its layers with layers of quantum neurons, effectively turning it into a hybrid *Quantum Neural Network*.

Our implementation of quantum neurons follows the approach of *Tacchino et al.* [9; 14]: A neuron that processes an input vector $\boldsymbol{i}$ of size m consists of $n = \log_2 m + 1$ qubits, which form a multi-qubit state $|00 \ldots 0\rangle$, except for one ancilla qubit. The multi-qubit state is then brought into an equal superposition of all the computational basis states using Hadamard gates. Subsequently, every computational basis state $|binary(k)\rangle$ additionally gets multiplied by $e^{i\theta_k}$, where θ_k is the k-th value in the input vector. This can be done using X-gates and multi-controlled phase shift gates $C_{n-1}R(\theta_k)$. We then want to form a scalar product with a similarly structured quantum state depending on the weights of the neuron's input, i.e., the network parameters that are being learned. To this end, we apply a similar series of gates "backwards": We first apply the Hermitian conjugates of multi-controlled phase shift gates $C_{n-1}R(\phi_k)$ and some X gates for the multiplication with $e^{i\phi_k}$, with ϕ_k being the k-th element of the weight vector $\boldsymbol{w}$, and then apply Hadamard gates and another "layer" of X-gates to obtain the desired superposition. Next, we apply a multi-controlled C_nNOT gate to "transfer the result" to the ancilla qubit. We then measure this qubit, obtaining 1 with a probability of $|\boldsymbol{i} \cdot \boldsymbol{w}^*|^2/m^2$. This last step represents the quantum neuron's activation function.

An example circuit for a 2-qubit neuron can be found in Figure 2. The time complexity of a network consisting of these kinds of quantum neurons would then be linear with respect to the size of the input vectors, $O(m)$ [9; 14].

Despite only requiring relatively few qubits per neuron, large neural networks such as the previously described PINN would still need several "typically entangled" [9] qubits. This need is still larger than what is currently available on quantum computers.

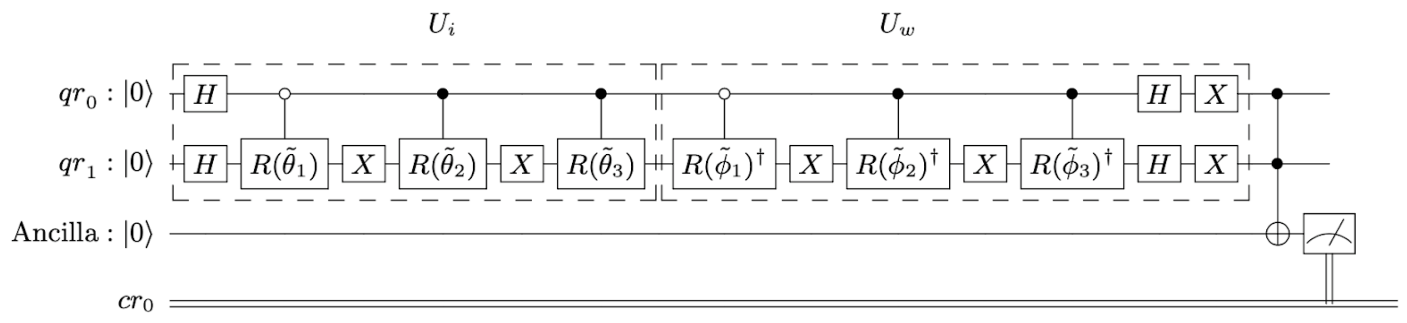


Figure 2: Proposed quantum circuit for our quantum layer [14].

Implementation

Computational fluid dynamics is all about solving partial differential equations numerically. The famous Navier-Stokes equation builds a set of PDEs of second order. Formation of shocks in fluid dynamics is well described by Burger's equation. Incompressible flow equations and continuity equations are PDEs. The Maxwell equations found in electrodynamics are also PDEs. To get faster and more reliable CFD simulations, we must start at the very beginning: solving PDEs. We thus provide a code base that solves PDEs leveraging classical and quantum PINNs as a proof-of-concept.

In a hybrid quantum neural network approach, it should be possible to circumvent the problem of limited available qubits by sequentially calculating the output values of individual quantum neurons, using the same qubits for each neuron. This would, however, lead to a significant decrease in the network execution speed. Moreover, errors in current quantum hardware would potentially diminish the solution quality [14]. Due to current performance bottlenecks (with regards to execution and solution quality), experiments involving the presumed quantum advantage of our proposition will have to wait until larger and more reliable quantum computers have been developed. Until this progress is achieved, we intend to use quantum simulators to determine the feasibility of our approach.

With the theoretical background in mind, we want to elaborate on our code. It is written with Google's *TensorFlow, Keras, TensorFlow Quantum, and Cirq* [15; 16; 17; 18].

In our code, we first implement a classical PINN that solves Burger's equation. We use a neural network mainly consisting of eight dense layers of width 20. In total, over 3000 parameters are to be trained, in 500-2000 epochs, depending on the desired accuracy. The results of this network were already demonstrated in Figure 1.
For the quantum approach, we define the new class QPINN and specify the quantum circuit proposed by *Tacchino et al.* [9; 14]. It is then converted into a tensor, so we can pass it to the network in an additional input layer. The circuit is displayed in Figure 3. The network architecture is similar to the classical network, except for the additional input layer and the quantum layer. Therefore, we also have over 3000 parameters to train. For the quantum layer, we use the built-in quantum simulator from *TensorFlow Quantum*. Currently, this approach does not yield a speed-up; therefore, it should be seen as proof-of-concept only. Thus, the number of training epochs is significantly smaller than in the classical approach, leading to reasonable execution times. With better and faster quantum computers available in the near future, we aim to provide a fast, fully functional, quantum powered CFD simulation software solution.
Since *TensorFlow Quantum* has been released only one year ago, some features are still missing. One of these features is a second order gradient for quantum circuits. An issue was already opened on Github, as many other researchers encountered this problem as well. Higher order derivatives are crucial for our PINN to work and we will thus have to wait until this feature is eventually implemented. To prove that our approach is working, we provide code for a quantum neural network in our repository using the circuit in Figure 3.

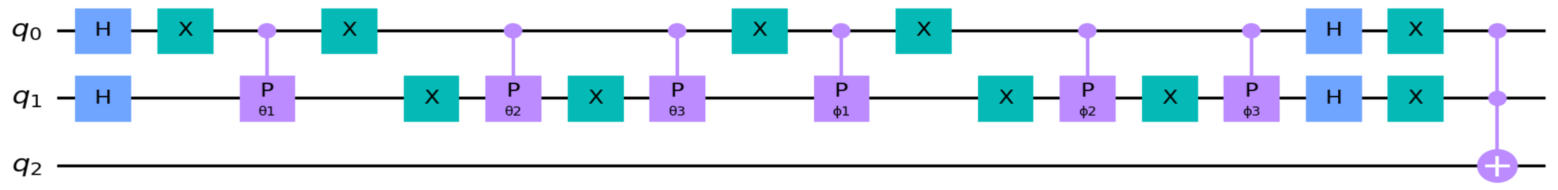


Figure 3: The quantum circuit for our quantum layer.

System Architecture & Roadmap

The product will be offered in both an API and UI-based option. The former will allow for direct connection to the backend systems through a REST API, while the latter would provide a user-friendly and relatively code-abstracted way of interacting with our systems.

Architecture

Figure 4 provides an overlook of the structure of our system. It consists of a layered architecture which contain several layers of abstraction that handle all the data traffic. The *backend* consists of our cloud servers (both HPCs handling general computations as well as Quantum machines that enable our hybrid algorithms). The *API implementation* and the *orchestration layer* will provide a scalable way of accessing our solutions either through the UI (*representation layer*) or directly by making calls through the API. The *caching layer* would allow for faster answer to common requests and reduce pressure on the backend resources. Finally, the *security layer* would ensure the required security and privacy of the data introduced in our system.

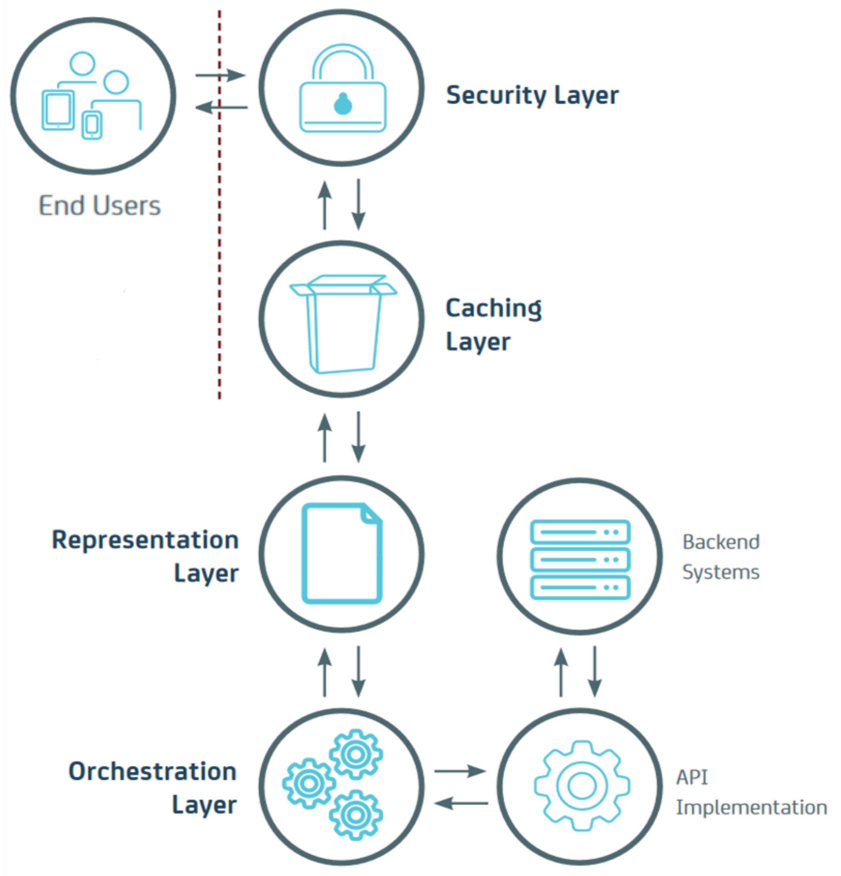


Figure 4: System architecture with relevant layers.

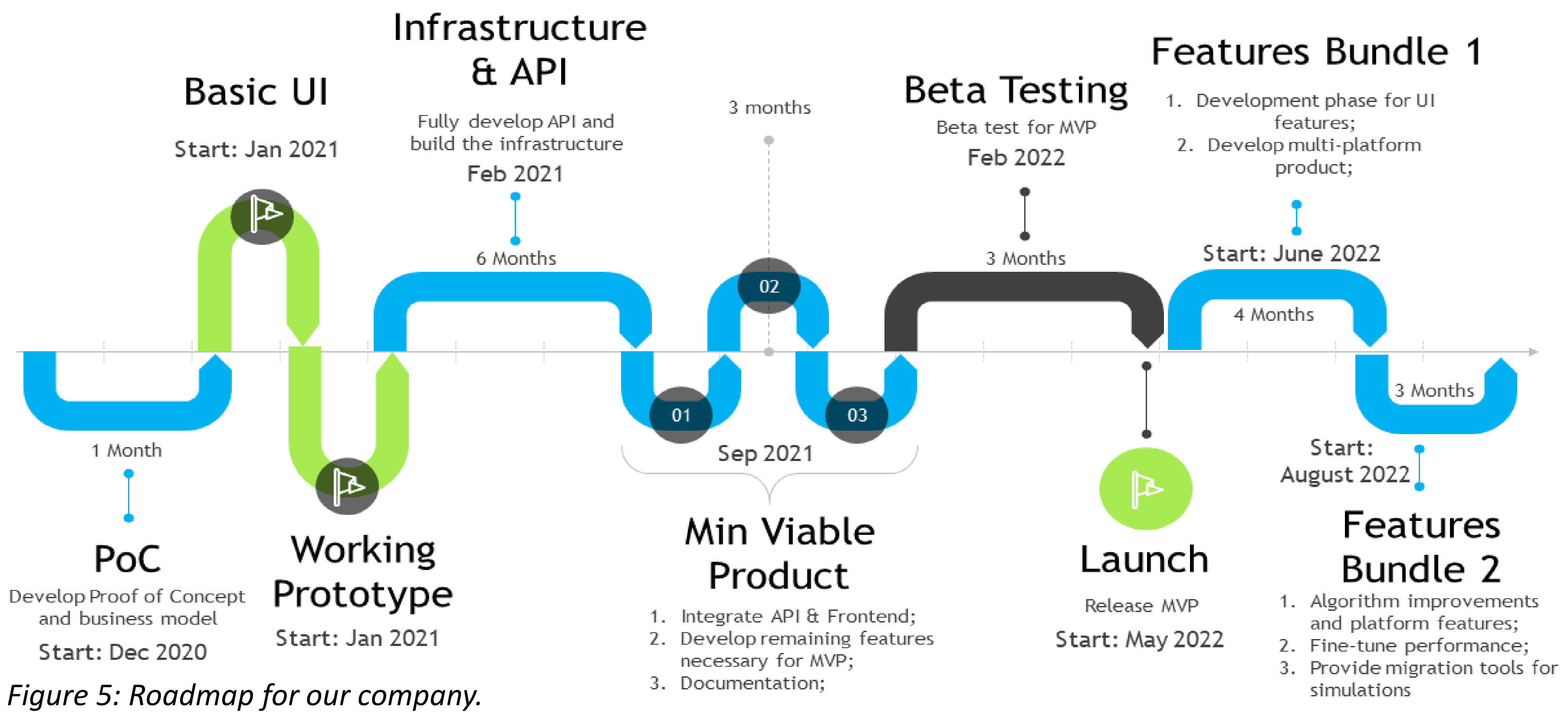
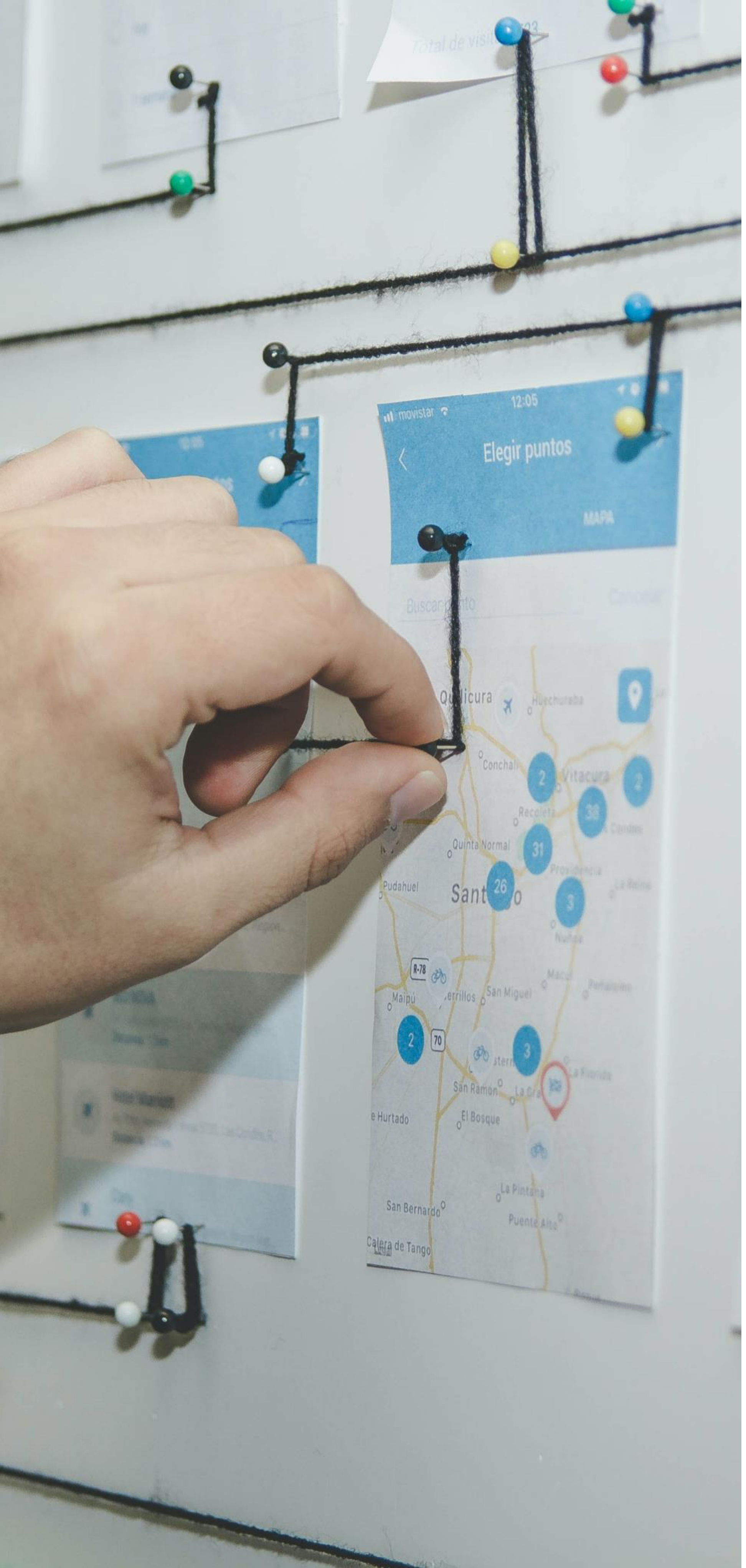

Infrastructure & API
Fully develop API and build the infrastructure
Feb 2021
6 Months
Basic UI
Start: Jan 2021
3 months
02
Beta Testing
Beta test for MVP
Feb 2022
3 Months
Features Bundle 1
1. Development phase for UI features;
2. Develop multi-platform product;
Start: June 2022
4 Months
1 Month
PoC
Develop Proof of Concept and business model
Start: Dec 2020
Working Prototype
Start: Jan 2021
01
03
Sep 2021
Min Viable Product
1. Integrate API & Frontend;
2. Develop remaining features necessary for MVP;
3. Documentation;
Launch
Release MVP
Start: May 2022
3 Months
Start: August 2022
Features Bundle 2
1. Algorithm improvements and platform features;
2. Fine-tune performance;
3. Provide migration tools for simulations

Figure 5: Roadmap for our company.



User Journey and Prototype

Overview of the main screen of the simulator UI.
The demo shows a simulation result being displayed, together with the available configuration available for the model.

The placement of the model navigation tools are also pinpointed on the right side of the user interface, together with the tools for exploring timesteps.

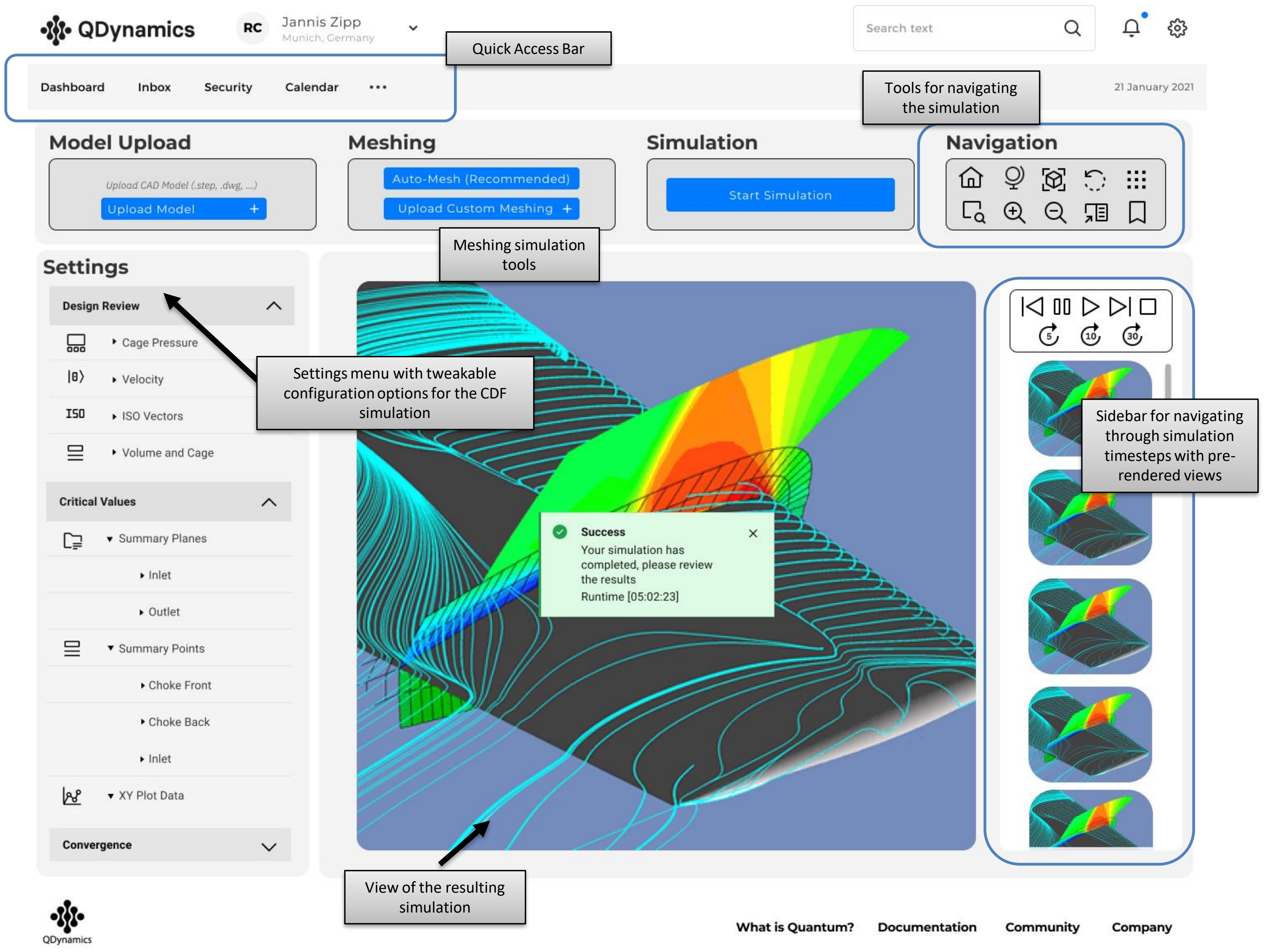


Figure 6: Commented user Interface overview. The complete, interactive UI prototype is available in Figma.

List of Sources & References

[1] Kyriienko, O., Paine, A. E., & Elfving, V. E. (2020). Solving nonlinear differential equations with differentiable quantum circuits. arXiv preprint arXiv:2011.10395.

[2] Computational Fluid Dynamics Market by End-users and Geography - Forecast and Analysis 2020-2024 (accessed last 02.03.2021) from https://www.technavio.com/report/computational-fluid-dynamics-market-industry-analysis.

[3] Langione, M., Kumar, A., & Tillemann-Dick, C. (2019). Where will quantum computers create value—and when. Boston Consulting Group, November.

[4] M. Raissi, et al. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, Volume 378, Pages 686-707. 2019.

[5] Montanaro, Ashley, and Sam Pallister. "Quantum algorithms and the finite element method." Physical Review A 93.3 (2016): 032324.

[6] Sirignano, Justin, and Konstantinos Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations." Journal of computational physics 375 (2018): 1339-1364.

[7] Dockhorn, Tim. "A discussion on solving partial differential equa- tions using neural networks." arXiv preprint arXiv:1904.07200 (2019).

[8] Narayanan, Ajit, and Tammy Menneer. "Quantum artificial neural network architectures and components." Information Sciences 128, No. 3-4 (2000): 231-255.

[9] Tacchino, Francesco, et al. "Quantum implementation of an artificial feed-forward neural network." Quantum Science and Technology 5.4 (2020): 044010.

[10] Grohs, Philipp, Fabian Hornung, Arnulf Jentzen, and Philippe von Wurstemberger. "A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations." arXiv preprint arXiv:1809.02362 (2018)

[11] Nguyen, Nam H., E. C. Behrman, Mohamed A. Moustafa, and J. E. Steck. "Benchmarking neural networks for quantum computations." IEEE Transactions on Neural Networks and Learning Systems (2019)

[12] Chen, Jialin, Lingli Wang, and Edoardo Charbon. "A quantum-implementable neural network model." Quantum Information Processing 16, No. 10 (2017): 245.

[13] Shao, Changpeng. "A quantum model of feed-forward neural networks with unitary learning algorithms." Quantum Information Processing 19, No. 3 (2020): 102

[14] Mangini, Stefano, Francesco Tacchino, Dario Gerace, Chiara Macchiavello, and Daniele Bajoni. "Quantum computing model of an artificial neuron with continuously valued input data." Machine Learning: Science and Technology 1, No. 4 (2020): 045008.

[15] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[16] Chollet, F. & others, 2015. Keras. Available at: https://github.com/fchollet/keras.

[17] TensorFlow Quantum: A Software Framework for Quantum Machine Learning arXiv:2003.02989, 2020.

[18] Cirq Developers. (2021, March 5). Cirq (Version v0.10.0). Zenodo. http://doi.org/10.5281/zenodo.4586899