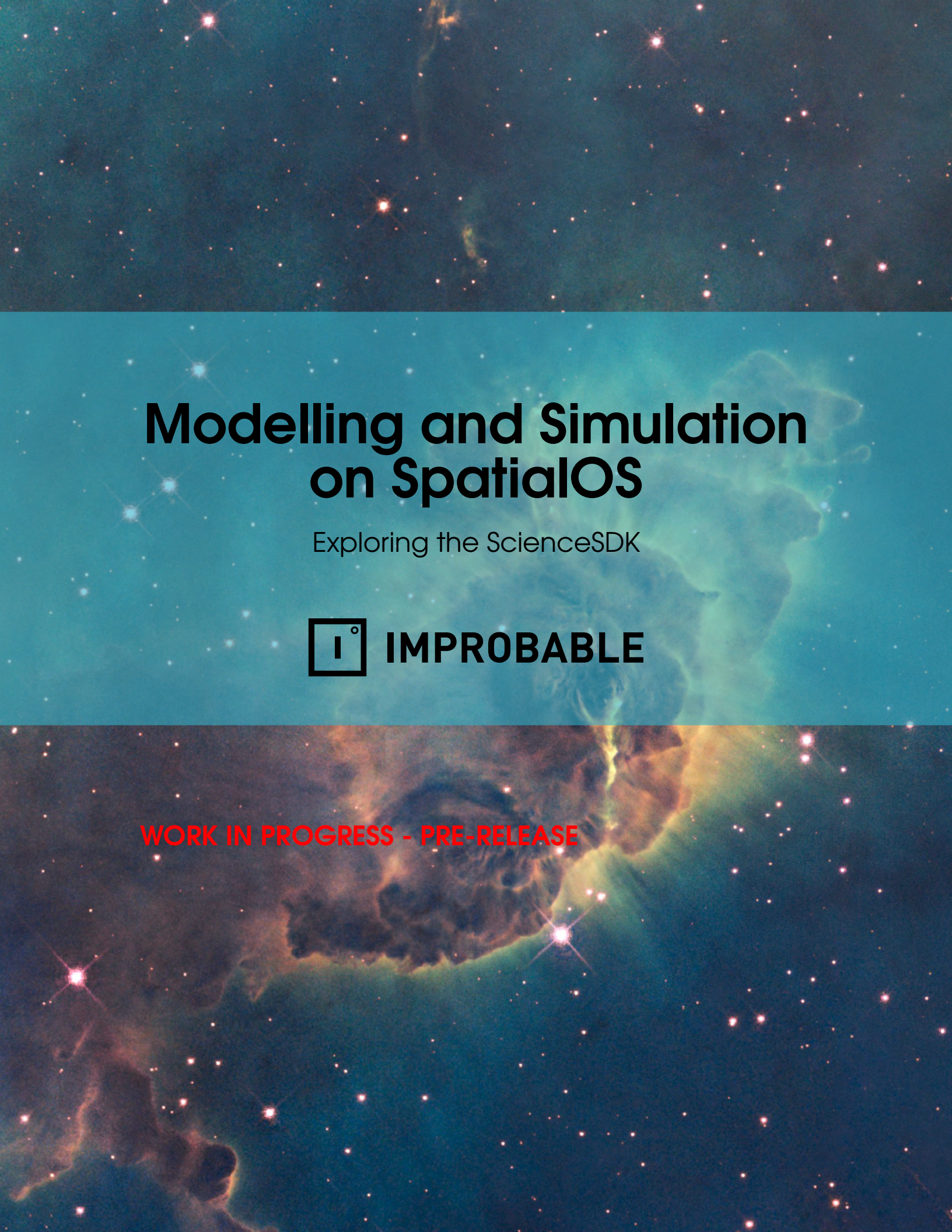


Modelling and Simulation on SpatialOS

Exploring the ScienceSDK

IMPROBABLE

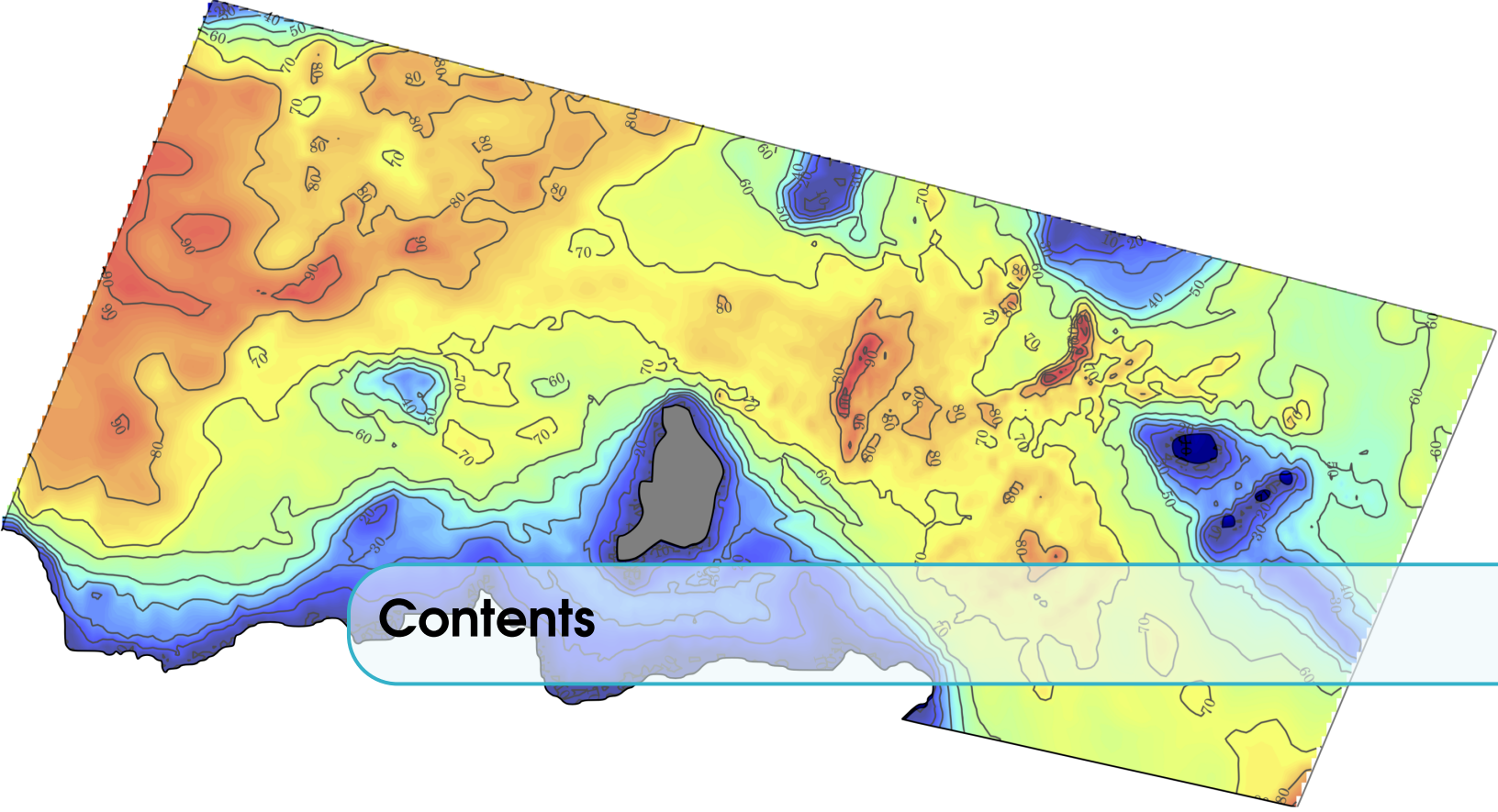
WORK IN PROGRESS - PRE-RELEASE

IMPROBABLE

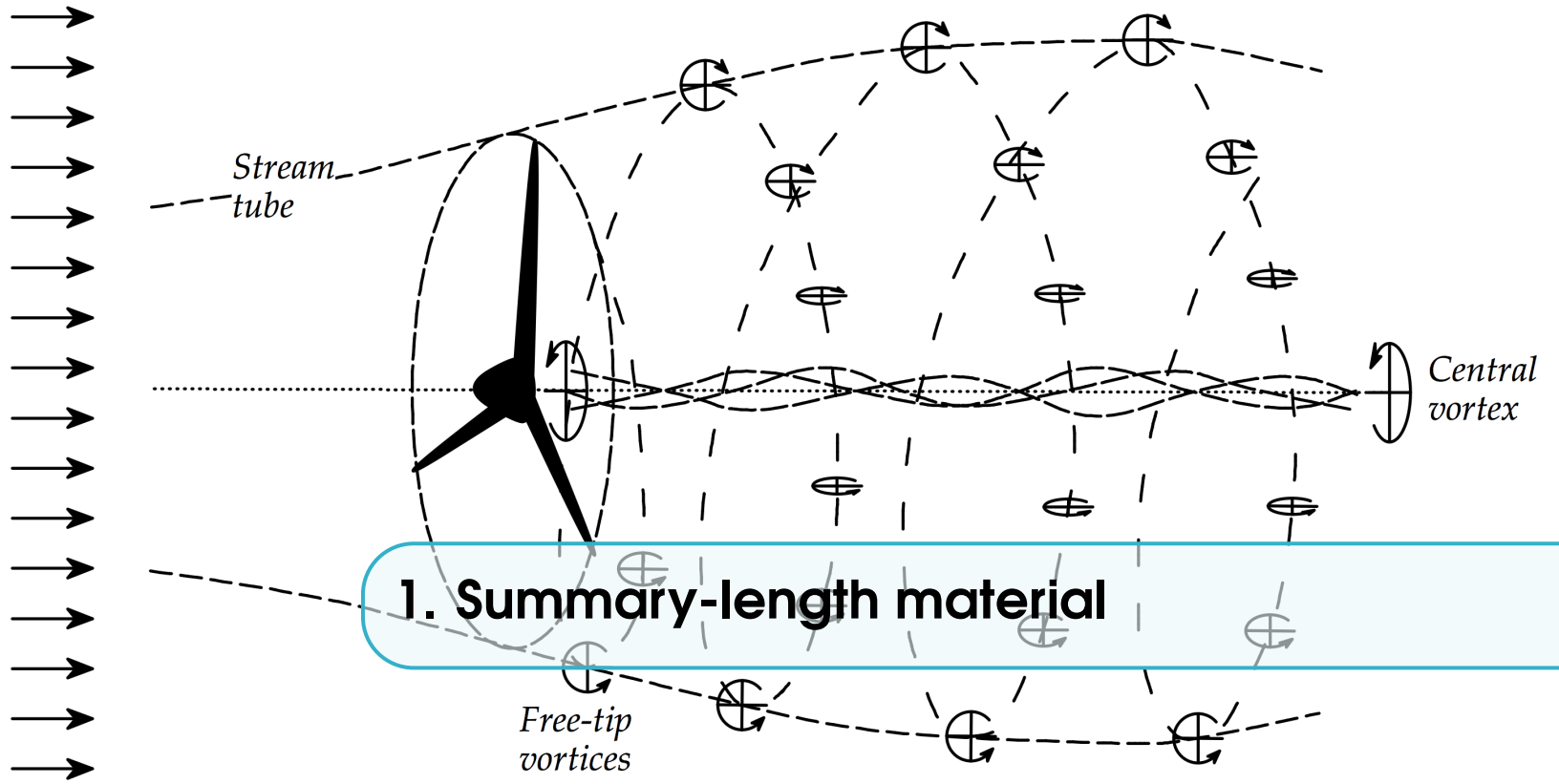
GITHUB.COM/IMPROBABLE-IO/SCIENCEOS

For more information contact daveculley@improbable.io

First release, 2017



Contents



1. Summary-length material

1.1 Introduction

Models are simplifications of real-world processes that enable us to make predictions about phenomena. The process of building a model involves using observational data to link (or 'map') inputs to outputs. Scientists and designers can then use this map to predict the system's response to a given input without having to rerun the experiment. These uses may involve the future of the planet, as in the case of climate change research, or the safety of hundreds of people, as in the case of building and aerospace design.

As the power and availability of computational resources increase with time, computational modelling has become a key cornerstone in the development of modern scientific understanding, where it is used in concert with laboratory and field testing. While real-world data can give isolated snapshots of phenomena and processes, it is the job of the computational model to firstly recreate this snap-shot (validation) then 'fill in the blanks' to give a more complete picture as to what activity is occurring.

1.2 An approach to modelling - Cheatsheet

1.2.1 The ten steps to modelling

Following these ten steps will ensure you consider everything you need to consider. In the examples presented in this booklet we will step through them explicitly – clearly the experienced modeller will do a lot of these steps implicitly. However, here at Improbable we have a specific product (SpatialOS) so there may be a tendency to put the cart before the horse (i.e. start with a modelling approach – spatial – and working back to a problem). Of course, an element of this is necessary to help our clients understand how we can help them, but it is worthwhile always having a structured approach to modelling in one's back pocket.

1. Problem formulation

- What is the problem
- What is the hypothesis
- Who owns the problem
- Who are the secondary, tertiary... stakeholders
- What is our role, what does success look like?

2. System identification and decomposition

- Inventory
  - Important (and relevant) concepts
  - Actors and objects (and how these relate to data we can observe)
  - Key measurements and performance indicators
  - Behaviours
  - States and properties
- Structure
  - States and behaviours of system components
  - Interactions between different system components
  - How behaviour at boundaries will be dealt with
  - How behaviours may be grouped into iterations, timesteps etc.
- Consider data availability – this will determine
  - Model fidelity
  - Flexibility of future model for calibration and validation
  - How model is structured to answer the problem

3. Concept and model formalisation

- Identify the class of model (governed by the problem and available data)
- Research how others have tackled similar problems previously (don't reinvent the wheel)
- Concretely define concepts in a rigorous framework
- Indentify appropriate data structures to codify concepts
- Define an ontology
- Model and parametrise behaviours and interactions formally (using maths / logic)

- Determine the model narrative using pseudo code (structure dependent upon the chosen time scheme)
- Don't forget input/output mechanisms often via observation operators

4. Boundary and initial conditions

- Models generally have dependecy upon previous state – good choice of initial state is vital
- Consider likely spin-up duration and behaviour
- Think carefully about how to model external influence at system boudnaries
- Be explicit about these treatments thay will likely strongly impact model results

5. Software implementation

6. Model verification

7. Experimentation

8. Data analysis

9. Model validation

10. Model use

1.3 Glossary of terms

We all come from different backgrounds – and our core tech comes from the games-sphere – and have picked up our modelling jargon from a diverse range of sources. It would be helpful if, within the division, we could develop a unified language with which to approach problems.

I think this would also be helpful for the wider company to interact with what we do on this side of the wall.

Tidy up the above

1.3.1 Model vs. Simulation

Model: A representation, or network of interconnected representations, of a concept/process or multiple concepts/processes. *Modelling a process implies you have discarded detail which is extraneous to the problem under-consideration and constructed a (computational) representation at an appropriate level of abstraction for the problem.*

Simulation: a model which has some proven legitimacy within some concretely defined scope to claim to represent (i.e. simulate) some real concept or process (ideally with some quantification of error and uncertainty). *Simulating a process implies you are augmenting your modelling effort with a good standard of calibration and validation. For a simulation, the level of modelling abstraction will be determined by both the problem & the available data.*

1.3.2 Time

Real-time: Time is represented in the model as it is in the real-world (albeit potentially sped-up) - consequently the progress of the model is dependent upon time. *Modelling a system which is active Mon, Tues & Thurs, but dormant on Wed - would mean the hardware sits idle for the Wed part of the simulation.*

Simulated time: Time in the model is dependent upon program progress. Methods such as discrete-time and constant step duration iteration are examples of the use of simulated time. *In the above example, we can use simulated time to ensure the hardware will skip over the idle Wed and jump straight to Thursday's work. Thereby we can solve problems as fast as the computer can compute, independently of problem's relationship to real-world time.*

1.3.3 Types of systems

Complex system: A system which displays emergence – features at the collective-scale which develop from the interactions of a large number of individuals at the unit-scale.

Complex adaptive system: A complex system in which the individuals at the unit-scale will adjust their behaviour dependent upon the features that develop at the collective scale. *Economics is the canonical complex adaptive system - the actors who comprise an economy will modify their*

behaviour in response to the emergent features of it.

Technical: Systems whose scope is non-behavioural - i.e. limited to purely scientific or technical processes.

Social: Systems whose scope is (almost) entirely behavioural.

Socio-technical: Systems in which the phenomena of interest involve the interaction of behavioural and technical behaviours / processes.

1.3.4 Types of model

Discrete event: A model which uses a discrete event treatment of time. Processors work through their task immediately, one after another – meaning that one processor may be ahead or behind another – there are various schemes which deal with this disparity in ways that may be more or less suitable for a given problem.

Discrete time: A time-stepping scheme whereby each iteration covers a small change in time and for each iteration there is a set of tasks that must be worked through.

Live-streaming: A model that is driven by real-world data - literally just a visualisation of the real world phenomena. *E.g. the map on the uber app - the position of the icons representing the cars is determined by the live-streamed position of the cars in the real world – there is no computation or logic used in their placement.*

Live-action: A model which (likely) runs in real-time and allows users to play-along to evaluate the consequences of a given action. *In non-deterministic models this evaluation will be crude (as repeating the computational experiment requires the 'player' to repeat the action over and over). SpatialOS (as a games platform) is clearly well suited to these. Note that validation and calibration is very difficult - one cannot be sure that the 'player' will behave in a manner that has been anticipated and for which the model has been validated.*

Speculative: A model which is, for all intents and purposes, un-validatable either because the data isn't available or because the scope of the problem is so wide that the range of possible outcomes is as big as to be impossible to prove valid. *This is not to cast aspersions on such a model's usefulness - indeed the competition to such a model may be a domain expert sat in a room saying 'I reckon' in which case such models will almost certainly provide insight.*

1.3.5 Modelling jargon

State: The current condition of the model, defined through the values of each of the model parameters. A capture of the model's state should behave like a checkpoint and contain all the requisite information to restart the model from the captured point. *In Spatial, 'snapshots' are used to capture the model state.*

Determinism: A given set of conditions will develop in the exact same way every time the model is run. A deterministic model will produce exactly the same answer every time it is run from the same

starting state.

Stochasticity: Randomness - generally a stochastic decision will be made by taking a random sample from a probability distribution with which we have represented a behaviour. *A stochastic model will produce a different solutions when run from the same initial state. The modeller must take steps to ensure a given solution returned by the model is in some way representative - and not an outlier. For example, a random sample of me asking what you had for breakfast yesterday will not necessarily equip me to make sweeping statements about your general breakfast habits - though of course it's better than nothing!*

Fidelity: Can be thought of a bit like resolution - how the unit-scale in the model compares to the way the problem plays out in the real-world. *We may model a pile of sand as a single object - a pile (low fidelity), or we may model each grain of sand (high fidelity). We could further model the pile at an atomic-level or smaller, the chosen model fidelity must be motivated by the problem being solved (and the available validation data in the case of simulations).*

Abstraction: The process of suppressing more complex details in order to focus on the mechanisms that are important to tackling the problem as posed.

Markov: Future states may be dependent on the current state but are not dependent on previous ones.

1.4 Glossary of concepts

1.4.1 Complexity and abstraction

While the ultimate goal of computational modelling may be to recreate reality in as accurate a way as possible, limits in computational resource render this an impossible goal. Rather, engineers must identify their objective, recognise the limitations of their computational resource and select models and strategies which (within computational budget) will help illuminate the behaviour of interest: For example, if we wish to model the physics of a bouncing ball, we do not need to consider it's colour, or what country it is in.

1.4.2 Hypothesis driven modelling

Since building tractable models therefore requires simplification, fundamentally the construction of models must be driven by their intended use. For scientists this is to provide specific insight, for designers this is to develop a product that is to be made. We model a phenomena based upon real-world observations but the complexity of the real-world precludes us from including all possible information in our model. In omitting certain information when constructing the model we, by definition, limit its utility. Understanding the simplifications we make when constructing a model is vital, as pushing the model beyond the point at which the simplifying assumptions hold, will produce inaccurate results (with potentially dire consequences - if, for example, we're using those results to design an airplane!).

When a model is used when simplifying assumptions no longer hold there may be two outcomes. In the best case, our model won't produce an answer – asking a physics model of a bouncing ball what colour the ball is just won't produce an answer. In the worst case, we will be misled – a model designed to show how commuters respond to a closed underground station might be tempting to use to study how travellers would respond to a bombed station, but the commuter's response to a bomb will be different to a routine closure, even though both relate behaviour to the inaccessibility of a given station.

1.4.3 Verification, calibration and validation

Building the model is the easiest and quickest part of modelling. Verification, calibration and validation ensure that the model we have built relates to the phenomena we were modelling in a way that generates the insights we were seeking. A model that has not been verified and validated generally has very limited utility (at best it may 'raise interesting questions' but there's no guarantee that those questions are reflective of the real-world scenario).

Verification of computational models simply ensures that the mathematical representation of the behaviours we intended to capture have been encoded accurately - that we haven't divided by two when we should have multiplied by two. This can generally be done mechanically using unit-testing and other code testing techniques.

In defining behaviours we may know the structure of the behaviour, but there may be free

parameters that need to be determined. Consider a thermometer, the mercury rises and falls with temperature, but ascribing the height of the mercury to the correct specific temperature is the process of calibration.

While calibration is data led - we tune parameters in the model so it better matches the data, validation is model led - we run the whole model (which comprises the set of parameters) and see if its output matches the observations. The difference is subtle, but validation checks the whole system is working together correctly and inoculates against over-fitting.

Consider our tube-system model, we might adjust our model to perfectly match a dataset we had from Balham station being closed one Sunday, then validate against separate data of Bank being closed. WIthout this second step, how can we be sure that we haven't set the behaviour to perfectly recreate the results when a small zone 3 station is closed on a Sunday - we may have over-fitted our model to specific behaviour unique to our one specific calibration data-set and thus compromised the generality of the model to capture other behaviours.

While often models are thought of as a product - it is built, validated, packaged and sold. In fact sophisticated models are living, cyclical things: Running the model raises questions - behaviour is observed, but is this behaviour a function of the model (i.e. due to the simplifying assumptions we've had to make) or is a function of the phenomena (i.e. the model is telling us something useful). So we improve and the model and validate the new set up, but as our understanding increases so we keep improving and validating. In this way a model is more like a city - it's never fully built but it is continuously improved; districts are rejuvenated, the city grows etc.

1.4.4 Model boundaries

It is inescapable when building complex models that we will find ourselves having to specify boundaries around the system of interest in order to make it tractable. For example, with our tube-network example, passengers will get on the tube at the Heathrow stations - this creates an interface with global air-traffic, which is linked to meteorology, which is linked to… which is linked to… Suddenly our tube network model has a global atmospheric model attached to it and the compute power required has increased by multiple orders of magnitude. Instead of modelling the Heathrow air traffic, we set the system boundary at the interface with the tube and simulate the contribution these passengers make by using a boundary condition - a cheap-to-compute simulator of the influx of passengers we expect from Heathrow. These boundary conditions must be carefully considered, poor assumptions and modelling of these can propagate through the model and compromise the results.

A special case of boundary conditions are initial conditions. Future behaviour generally has some link to current state - in the real world, our current state has evolved slowly from the start of the universe, as modellers we don't have time to go back that far! Thus we have to pick a sensible starting position and potentially give the model a chance to 'spin-up'. For our tube model, perhaps we start the model at 3 a.m. with no passengers - most of the lines are closed and those that are open will be fairly empty naturally. To get an accurate simulation of night-time tube usage we would then need to wait to the next simulated night, when the initial condition of no usage the night before has had a chance to average out.

1.4.5 Model coupling

Clearly if Heathrow will have a model of how frequently flights land and take off, it would seem natural just to couple it to the tube-system model. In practice this is very difficult due to differences in the nature of the systems being coupled the models will generally have different architectures, abstractions and be built to supply different insights. This is something we hope to make easier with ScienceOS.

One thing to be careful of here is that as models are coupled as modules in a larger simulation, the model complexity increases very rapidly. Assumptions made by the designers of the 'other' models must be understood. The way those assumptions will interact with the simplifications in other modules will also need to be understood. Validation is, again, key.

Timestepping, fidelity, spatial resolution etc.

1.4.6 What can we do with models

2nd order effects Emergent behaviour is a term used to describe complex system-level behaviours that emerge from relatively simple component-level ones. For example, the economy comprises a host of individual actors who (according to the capitalist model) act in their self-interest this causes surges and recessions at the world-economy level (each actor responds to the emergent macro-level behaviour of the economy and adapts their behaviour). In these complex adaptive systems, additional effects may also develop. Every action has an effect, and those effects have consequences – picture a line of dominoes, we only push the first one but the result is that the last in the chain is knocked over.

In rich, complicated models it is usual to see these causal chains develop, in fact this is one of the key reasons we have interest in, and develop such models. However, we must exercise caution - if the new behaviours that develop are not within the scope of the modelling assumptions that have been made, they will not have empirical validity. For example, returning to our example of a tube station closure, we may notice that many people circumvent the closure by walking all or part of their journey, leading to an large increase in pedestrian traffic. In reality of course this is true - to an extent - but people will also catch buses and cycle - if these behaviours aren't encoded into our simulation then we may draw the wrong conclusion - that everyone walks. Clearly this is a trivial example, but in rich models it is an important consideration to keep in mind. The correct way to handle second order effects in this situation is to note 'It looks like this has the additional effect of massively increased pedestrian traffic' see that this is worth investigating then auditing the model you have to determine 'is this model still suitable for investigating increased foot traffic due to tube closure'. In this case we will conclude that bus riding and cycling are addition behaviours that must be considered and encoded (and validated) to answer such a question.

1.4.7 Determinism and modelling under uncertainty

Deterministic models are those which consistently and repeatedly produce the same results. A model of a bouncing ball could reasonably be modelled deterministically, within some negligible margin of error repetitions of the dropping experiment in real-life will produce the same result, so my model of

it should also produce the same result. Consider instead dropping a rugby ball, infinitesimally small adjustments in the way we drop the ball will result in very large differences in where the ball ends up after its first bounce. Using a deterministic model to predict the behaviour of the latter would provide little useful insight.

In complex and dynamical systems, in which non-linear and multi-scale processes interact, uncertainty can exist in many forms. Often uncertainty is introduced through the presence of randomness (for example turbulent eddies in fluids) that is inherent to the (physical) processes being modelled, and is impossible to describe in a deterministic way. While sometimes it arises because there is an intrinsic lack of knowledge relating to either the process or the parameters being modelled. Correctly dealing with these uncertainties is a significant challenge.

One of the frequent shortcomings of computational modelling in practice is that, while the outputs that are being provided to decision-makers are fairly simple (for example a plots and graphics) they bely great complexity in the underlying model and there may be great uncertainty in the information and data from which they were generated. Compounding this, computational modelling is often an intricate procedure, and those who develop and run the models are generally not the same as those who use the results to make project decisions. Uncertainty in the inputs, randomness in the physical processes and limitations of the models (either from wrong / incomplete equations or discretisation errors) are often not communicated alongside the results, meaning that it is all too easy for this uncertainty to be (inadvertently) neglected and for it to go unaccounted for. Simply put, it is imperative that results which are to be used to determine the viability of – and make design decisions on – a multi-million pound industrial project are presented with appropriate ‘error-bars’.

1.4.8 Visualisation

Models themselves are generally hidden to all but the people who build and maintain them. With smart enough communication of the results it is very easy to get buy in from people who are either not thinking critically or who do not possess enough domain expertise to be analytical. This makes it easy to accidentally miscommunicate or intentionally manipulate those people. Visualisation is a specialist subject area in its own right.

One crucial question is; as models tend toward the appearance of representing the real world, how do we communicate model results in such a way that makes clear the limitations of the simulation?

Defining the problem It would be good to condense the discussions that will ensue over the next few weeks (while we pick a problem we use internally) into some sort of problem paradigm that we can get our hands around, communicate to customers and for which we have concrete examples and helpful metaphors.

2. A tao of complex modelling

2.1 Introduction

2.2 Problem formulation

Modelling is a means to an ends – rather than an end in and of itself. The goal of most models is to provide insight into the possible nature of a real-world system. They allow the modeller run ‘in silico’ experiments that would be difficult or expensive to run ‘in vitro’.

Models are tools to help us understand the dynamics of systems, explore possible futures or experiment with scenarios and decisions to determine whether they yield desirable or detrimental states.

In pure science, addressing the lack of insight is the goal, in technical systems – engineering applications – we use the insight to inform design decisions and in social applications we use the insight to inform policy design.

If modelling exists to address deficiencies in insight, then the process is necessarily goal oriented, and well formulated problems are more likely to yield useful insights. Thus, the definition of well formulated problem lies at the heart of a successful modelling strategy.

2.3 System identification and decomposition

The next step in approaching the model is to formulate a generativist description of the problem. This means the system must be identified and bounded, then decomposed into the constituent components that we will need to represent in order to simulate the behaviour of interest of the system under anaylsis.

At this stage we're not looking to formalise the representation of the system or its constituent components – that is the next step – resist the temptation yet to think about *how* you will model things and focus instead on *what* you will need to model in order to tackle the problem as it was defined in the previous step. Hopefully this illustrates the importance of the previous step, if the problem identified is "how high will this ball bounce" we can, at this stage decide that we need not capture in the model that the ball happens to be red.

It is at this stage that we, as modellers, need to seek expert input. It is not reasonable to expect us to know everything (or indeed anything) about every problem with which we are presented. Of course our modeller's intuition will invariably feed in at this stage, but this should be augmented and verified through conversations with domain specialists and other modellers (who may have differing backgrounds and thus a different intuition) and reading of academic papers, websites etc. etc. Remember that there is nothing new under the sun, and the overwhelming liklihood is that someone will have tackled this, or a similar problem, before – take every opportunity to learn from their conclusions..

This step is concerned with **inventory** and **structure**. The former is concerned with identifying the boundaries and components of the system and their important states etc.

Inventory - identify the following:

- Important (and relevant) concepts
- Actors and objects (and how these relate to data we can observe)
- Key measurements and performance indicators (linked to the problem!)
- Behaviours
- States and properties

The structuring phase is then concerned with how the components interact with one another and the system boundary, and how we may most appropriately handle spatio-temporal discretisation.

Structure - consider the following:

- States and behaviours of system components
- Interactions between different system components
- How behaviour at boundaries will be dealt with
- How behaviours may be grouped into iterations, timesteps etc.

Much of the goal of this step is to consider the appropriate model fidelity. This encapsulates a host of factors that must be appraised, and this process will, to some extent bleed through into the next modelling step.

In addition it is worth, at this stage, thinking about the data you have (or are likely to be able to get), is the modelling fidelity appropriate to the fidelity of the data? How will the model be structured to facilitate validation? What sources of uncertainty are likely to exist and how will we minimise, quantify and communicate it?

2.4 Concept and model formalisation

Having identified a conceptual framework for the system and its components the next step is to formalise it. To a large extent, the way in which we choose to formalise the concepts will be dependent upon the class of problem with which we are presented and the modelling paradigm we decide to employ. For example, to study the migratory behaviour of herds or flocks of animals we may model each individual animal as a decision making agent, or we may model the whole group as a continuum – with a spatially varying density as a function representative of the local density of animals. Both approaches are valid, with the degree to which they are suitable contingent upon the problem being addressed and the fidelity of the calibration and validation data available.

Now that the overarching modelling strategy has been determined, it is time to concretely define the concepts, agents, objects and behaviours in a rigorous framework. This will generally involve an amalgam of mathematics and pseudo-code. For uncertain or stochastic parameters, decisions must be made on the appropriate (frequency / probability) distributions to deploy and in general, judgements should be made on suitable parametrisations and efficient data structures to represent the concepts that have been identified.

An important element of this stage is to define a governing ontology for the model – that is a formal naming convention for all elements (agents, objects, properties, behaviours...). There are a few things to bear in mind here: Firstly it is generally best to adopt and adapt conventions from the existing literature; unless there is a compelling reason, there is little benefit in reinventing the wheel. This facilitates easier communication of concepts with domain experts and may enable easier coupling and comparison of models in the future. Secondly this will become an important mechanism for linking real-world data with its modelled counter-part. This integration will form a foundation for our calibration and validation operations in due course.

Remember to always link back the problem at hand to ensure that the model, as constructed, will actually answer the pertinent questions as asked and provide pertinent insights. An important part of this is to think about the experiments that will be run using the model and the observations that will be made on those experiments. Observation operators need to be defined, and the mechanism by which they link to (and are compared to) the real-world data formalised. How will uncertainty in the real-world data be integrated into the model with it? Does the available data relate to something directly modelled or some kind of hidden or proxy element?

3. Modelling on ScienceSDK

3.1 Introduction

3.2 Structuring models for the ScienceSDK

Given the intent of this booklet as an introduction to modelling on the ScienceSDK, let's look more closely at how to formalise our model to best suit the structure of that tool. See chapter XXX for an introduction to the ScienceSDK. One of the core concepts of the ScienceSDK is to present to the modeller a familiar object orientated programming paradigm (rather than the message passing paradigm of SpatialOS which is likely to be less familiar to the modelling community). Consequently, the key programming unit in ScienceSDK is the *Migratable*. A *Migratable* is an object which is capable of being moved around in entirety to different processors. *Migratable* objects are constructed with corresponding *Reference* objects which are constructed for the user by the ScienceSDK. *Migratables* refer to one another indirectly via the *Reference*s, which provide pointers to the objects themselves, on whichever processor they may currently be residing. A migratable is consequently the smallest unit of code in the ScienceSDK paradigm.

So how can we structure concepts and models around this paradigm? Well the obvious mapping is the tangible agents and objects to migratables, with more abstract objects – for example observation operators – also treated similarly. The spatial paradigm is an intrinsic characteristic of the underlying SpatialOS treatment of objects (spatial location is a key characteristic used for load-balancing) but ScienceSDK abstracts this away from the user – where it can be appropriately applied to a migratable it may be defined by the modeller, where inappropriate it shouldn't be defined, allowing the ScienceSDK to load-balance more effectively.

4. Solar system

4.1 Introduction

Physicists suspect that a large proportion of the mass in the universe (the mysterious dark matter) is bound up in tiny black holes which were formed during the big bang. Over time, these black holes have shrunk due to Hawking radiation but there may still be primordial black holes (PBHs) of up to 10^{26}g (around the mass of Mercury) passing through the universe at speeds of around 300 kms-1. Physicists have considered the effect of a smaller PBH striking the Earth but here, we'll consider a more Improbable World: the effect of a 10^{26}g PBH entering our solar system on a possible collision course with Earth.

Since black holes are...erm...black, we couldn't directly observe the position and velocity of the PBH so calculating whether it really is on a collision course (and whether we should continue paying into our salary sacrifice pension) wouldn't be easy. Luckily, a PBH that massive would potentially perturb the orbits of the other planets so by watching the wobble of the planets we can work out the position and velocity of the PBH and so work out if it is likely to hit the Earth.

4.1.1 Modelling goals

- Create a physically realistic model of our solar system
- Use observations of the position of Mercury to calculate the position and velocity of a passing PBH and work out if it is on a collision course with Earth

4.1.2 Take-home insights

TODO: based upon the position of this tutorial in the series and the preceding lessons.

4.2 Problem formulation

4.2.1 What is the problem

The core problem is: Will the Earth be hit by a primordial black hole? An ancillary problem is that PBH's cannot be directly observed, and as such their position must be inferred by their effects on the massive bodies they may effect.

4.2.2 Hypothesised approach

The mechanics of orbiting bodies are well understood, and can be modelled accurately with relative ease. In this scenario we are astronomers who want to build a model of the solar system which is able to detect planetary wobble due to a primordial black hole (PBH). So the system we build must be able to compare its own output to observations made in the real world and detect discrepancies due to planetary wobble. Given this wobble we need to be able to estimate the location of the PBH at a given time, then add this to the model and run it forwards into the future to determine the likely trajectory of the PBH (given the errors in estimating its location in the first place) and ultimately compute a probability of collision with earth.

- Build an accurate model of the dynamics of the solar system in 'normal' conditions.
- Monitor the difference between the observed position (i.e. the position as observed by real-world telescopes) of Mercury and its position in the model.
- Estimate the position of the PBH and bound the uncertainty attached to that estimation.
- Estimate the trajectory of the PBH and the probability that it will collide with Earth.

4.2.3 Problem owner and additional stakeholders

Since this is a purely technical system, we do not need to consider whose problem we are addressing, and the other stakeholders and actors whose behaviour and motivations would be an important dynamic in a social or socio-technical system.

4.2.4 Our role

Our role, as modellers, is to extract knowledge from domain experts to build a model which addresses the problem as formulated in bullet points above. Perhaps most importantly, it is to communicate the assumptions under which the results of the model will hold alongside those results.

4.3 System identification and decomposition

Thankfully, we all studied and remember A-level physics (!) and therefore have a good amount of domain knowledge already at our fingertips. In this case, the system boundaries are (unusually) well defined – we wouldn't expect any interaction between objects inside and outside the solar system - so we can limit our modelling to that extent. The main system components are the planets, the PBH

and the Sun - these can be grouped as 'massive bodies' (i.e. bodies that have mass). My intuition, at this stage, is that satellites are unlikely to play a big role in the problem (Mercury has no known satellites – and by the time the PBH is close to Earth, it's too late for the Moon to have much impact). Of course a rigorous modelling approach would support this assertion with appropriate computation.

The important states of our massive bodies are their location, $\vec{x}(t)$, and velocity, $\vec{u}(t)$. Massive bodies interact via the gravitational force their mass exerts, we recall (!) that the gravitational force, $\vec{F}_g$, between two bodies with masses m_1 and m_2 separated by $\vec{r}$ is

$$\vec{F}_g = -\frac{Gm_1m_2\vec{r}}{|\vec{r}|^3}. \tag{4.1}$$

We further recall (!) that an unbalanced force acting on a body will induce an acceleration

$$\vec{F}_g = m\vec{a}, \tag{4.2}$$

and that this acceleration can be used to update the velocity and position of the body. Thus, $\vec{a}(t)$ is an intermediate property of our massive bodies that is important to keep track of, and the seperation distance between pairs of bodies is an important factor controlling the extent of their interaction.

In this case, timestepping would seem a sensible temporal scheme, we define a timestep length, Δt (often just called dt), at each iteration we'll step forward in time, compute the resultant gravitational forces acting on each body, compute the acceleration induced on each body, update the velocity of each body and finally compute the new position of each body. Note that this temporal discretisation induces an error (a 'discretisation error' no less) – we are assuming that the acceleration of a body is constant for the duration of the timestep, where patently this is false. We can ultimately measure the impact of this error by exploring the effect of adjusting dt in our final model.

4.4 Concept and model formalisation