Institut für Informatik
der Technischen Universität München

Lehrstuhl für Informatik mit Schwerpunkt wissenschaftliches Rechnen

---

Multiscale calculus with applications in quantitative finance

Stefan Dirnstorfer

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universiät München zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigten Dissertation.

Vorsitzender:		....
Prüfer de Dissertation:	1.	....
	2.	....
	3.	....

Die Dissertation wurde am 23.08.2005 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am .... angenomen.

Acknowledgement

The autor wants to thank his professors Hans Bungartz, Rudi Zagst and Christoph Zenger for supervising this thesis. Special thanks to Michael Ege for his contibutions to this document, his criticism and the many discussions on practical implications. The author feels much obliged to Andreas Grau for proof reading the document and for the many discussions on possible applications. Arnd Pauwels has helped the author in questions of mathematical rigor.

Contents

0.1 Introduction . . . 1
   0.1.1 Algorithms on the functions domain . . . 1
   0.1.2 Layered model . . . 3
0.2 Summary of each chapter . . . 4
0.3 Operator index . . . 5

1 Multiscale Calculus **6**
1.0 Introduction . . . 6
1.1 Multiscale functions . . . 7
   1.1.1 Axioms . . . 7
   1.1.2 From real functions to multiscale functions . . . 9
1.2 Computer implementation . . . 11
   1.2.1 Java . . . 11
   1.2.2 C++ . . . 12
   1.2.3 Maple . . . 15
   1.2.4 Other languages . . . 17
1.3 Basic calculus . . . 18
   1.3.1 Differential . . . 18
   1.3.2 Integral . . . 21
   1.3.3 Fundamental theorem of multiscale calculus . . . 25
1.4 Advanced calculus . . . 27
   1.4.1 Solver . . . 28
   1.4.2 Blur operator . . . 29
   1.4.3 Continuous shift operator . . . 31
   1.4.4 Convolution . . . 32
1.5 Adaptivity . . . 33
   1.5.1 Interpolation . . . 33
   1.5.2 A priori adaptivity . . . 34
   1.5.3 A posteriori adaptivity . . . 35
   1.5.4 Sparse grid . . . 36
1.6 Proxy operators . . . 37
   1.6.1 Cache . . . 37
   1.6.2 Multiplexer . . . 38
   1.6.3 Other proxies . . . 38
1.7 Conclusion . . . 39

2 Pattern in portfolios **41**
2.0 Introduction . . . 41
2.1 The economic state . . . 42
2.2 Valuation function . . . 42

2.3 Processes . . . 43
2.3.1 Discrete process . . . 44
2.3.2 Brownian motion . . . 44
2.3.3 The Black&Scholes model . . . 44
2.4 The Portfolio . . . 46
2.4.1 Transaction . . . 46
2.4.2 Option . . . 47
2.5 Pricing via arbitrage . . . 50
2.5.1 The process . . . 50
2.5.2 Arbitrage-free price . . . 51
2.5.3 Equivalent martingale measure . . . 51
2.6 Example . . . 52
2.6.1 Hedging activity . . . 53
2.6.2 Supply and demand . . . 53
2.6.3 Market impact . . . 54
2.7 Conclusion . . . 55

3 The Theta-notation for stochastic processes 56
3.0 Introduction . . . 56
3.1 Stochastic model . . . 57
3.1.1 Transition probabilities . . . 58
3.1.2 Deterministic transition . . . 58
3.2 Differential processes . . . 59
3.2.1 Drift operator . . . 60
3.2.2 Blur operator . . . 62
3.3 Stock price models . . . 67
3.3.1 The Black&Scholes model . . . 67
3.3.2 GARCH . . . 68
3.3.3 Copulas . . . 70
3.4 Term structure models . . . 71
3.4.1 Deterministic model . . . 71
3.4.2 Heath-Jarrow-Morton . . . 72
3.4.3 Hull&White . . . 73
3.5 Example . . . 75
3.5.1 Present value . . . 76
3.5.2 Risk measures . . . 77
3.5.3 Stopping times . . . 77
3.6 Conclusion . . . 78

4 Put into practice 80
4.1 Economic model . . . 80
4.1.1 The product . . . 80
4.1.2 The process . . . 81
4.2 Algebraic results . . . 81
4.2.1 Simplified setting . . . 81
4.2.2 Integral form . . . 82
4.2.3 Logarithmic scale . . . 83

4.3 Numerical solution 83
  4.3.1 Integration 84
  4.3.2 Standard scale 85
  4.3.3 Logarithmic scale 85
4.4 Further tuning 86
  4.4.1 Evaluation nodes 87
  4.4.2 Local adaptivity 88

5 Conclusion **90**

About this document

0.1 Introduction

This document establishes a concept for a computer implementation of a function data type on which mathematical operators can be implemented. The functions should have the flexibility to allow for common operatations such as integration, differentiation, inversion and the solution of differential equations. Furthermore its flexibility should extend to the algorithmic concepts adaptivity and sparse grids. Chapter 1 shows how to implement such a data type and how the common mathematical operations work on it. The algorithms are presented in a discrete algebra and as computer code. Chapter 2 shows the transformation of economic investment strategies into functional operators. Based on functions as basic data type the evaluation of typical problems in quantitive finance is kept compact and short. Chapter 3 goes into the details of specific models for the dynamics of economic parameters. Many well-known processes from quantitative finance are turned into the modular and computer implementable operator form. The final chapter 4 puts everything into practice and demonstrates the applicability of the concept for mathematical modelling as well as for numeric implementation.

0.1.1 Algorithms on the functions domain

Since the very first days, computer science was driven by the search for solutions to mathematical problems. It started with basic arithmetics on the domain of integer and floating point numbers, and evolved into the algorithmic machinery to execute well defined sequences of calculations. With the appearance of FORTRAN, the universal formula translation language, it could well be claimed to have implemented all tasks of linear algebra, based on the array representation of vectors and matrices.

With growing computational resources another mathematical discipline is now in the reach of computer programmers. It is the field of calculus, which deals with functions and functional operators. As much as a vector is a collection of finitely many numbers, a function has infinitely many values and must be approximated appropriately. While the vector dimensionality refers to the number of vector components, the functional dimensionality is the number of independent function arguments. High dimensional functions cause various computational difficulties associated with the “curse of dimension”.

Example: Suppose we wanted to write a simple iterative solver for an iniatial boundary value problem. The problem is known in finance as the pricing of an american option and will be discussed in chaper 2 in more detail. Just by looking at the PDE below we can single out the discontinuous term $1_{f<x-K}$ that certainly prevents this equation from beeing solved by standard PDE software.

$$\frac{d}{dt}f = \frac{\sigma^2 \mathrm{x}^2}{2}\frac{d^2}{d\mathrm{x}^2}f - \mu \mathrm{x}\frac{d}{d\mathrm{x}}f + 1_{f<K-x} \tag{0.1}$$

However, the characteristic function only prevents the function f from falling below $K - x$, as it implements a large positive drift where the value of f falls short. This finding allows for an efficient solution of this problem, as it consists only of convection diffusion term with analytic solution and a maximization to ensure the minimum. Figure 1 presents the code for an iterative solution of the diffusion and the pointwise maximization in discrete time. Such programs can easily be entered into computer algebra systems. However due to the nonsmooth maximazation one is very unlikly to obtain a simplified result. Out of the box numerical packages will have difficulties to interpret functions as data types and would require significantly longer program codes.

```
FUNCTION mypde(f)
   FOR i := 1 to 100
     f(x) := $\int_{\mathbb{R}}$ EXP($-\frac{1}{2}\left(\frac{x-y}{\sigma}\right)^2$ - $\mu$) f(y) $dy$
     f(x) := MAX( f(x), K-x)
   END
RETURN f
```

Figure 1: This document aims for an algorithmic framework in which functions can be used as normal data types. The function `mypde` demonstrates an example that requires the flexibility of computer algebra systems and the power of advanced numerical methods.

Available concepts Functional algorithms have since been developed for the representation of functions, the solution of integrals, differential equations and variational problems. Good algorithmic concepts are available and have been implemented to solve very specific problem sets. All current approaches have the same shortcoming of either having limited scope in their applicability or

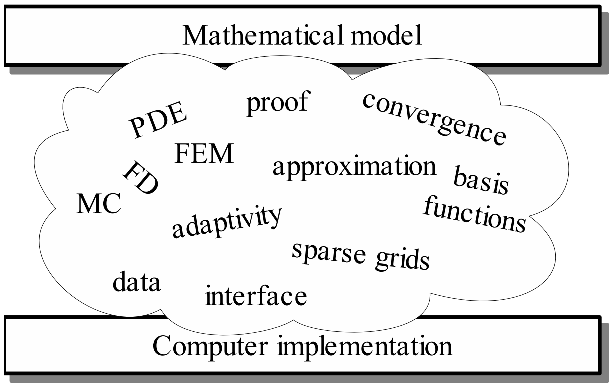


Figure 2: The computational solution of mathematical problems on the domain of functions is hardly supported by standardized data structures or programming interfaces.

being too general without significant added value over plain vector algebra. Everybody who wants to solve a new mathematical problem that involves the variation of functions and does not fit precisely into the domain of existing packages must start from the beginning. There is no such thing as a general data structure for a function that allows for standardized and clear algorithms to solve equations, optimize the value or perform convolutions and differential calculus.

0.1.2 Layered model

Calculus, or the mathematics of functions, provides one of the most advanced notations for mathematical models, as we know them from physics and economics. The notation is compact and allows for simplifications and transformations. For the translation of calculus into software this document suggests two intermediate steps to be taken. First, the mathematical problem must be represented in an explicit notation, that is compatible with the vocabulary of common computer algebra systems. This requires all computational steps to be explicit instead of being hidden in prose and text forms. Furthermore all domains must comply to the basic preliminaries of computability, i.e. sets must be finite or have a norm that allows finite approximation. The second step concerns the representation of the algorithm and its modules. Extremely complex algorithms might work in some instances, but can not be used as a mathematical tool to be used in untested environments. This document suggests a computational calculus that leads to a direct implementation in a functional programming language. It can be turned into efficient code in a separate step.

Explicit notation Before a computational task can be implemented we want it to be represented in a computer algebra compatible notation. Thus the numeric challenge should be explicit without having to read any informal and possibly ambiguous prose. Some of the computational steps involve the solution of equation systems and require specific operators, which only have an implicit

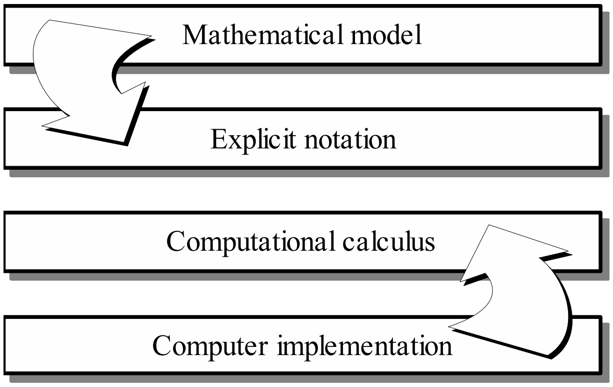


Figure 3: This document suggests a computational calculus and an explicit operator notation that simplify the transition from math to code.

definition via their properties. These operators must then be given an explicit approximation strategy in computational calculus.
Chapters 2 and 3 introduce the explicit operator notation in quantitative finance. Prevailing financial literature, despite large terminology, does not have a mathematically precise and welldefinede notation. The operator notation provides a straight forward implementation procedure for the problems of financial institutions.

Computational calculus The goal of computational calculus is to make the algorithmic behavior explicit. The way a numerical system deals with limits, integrations and function inversion must be reasonably traceable without having to study endless program code. The ideal computer calculus has a simple notation that directly leads to an effective program, and can be implemented in an efficient fashion.
Chapter 1 uses multiscale calculus to define functional algorithms. It is based on a standardized data structure for functions. The structure is compatible to arrays but automates the specification of domain borders and sample spacings.

0.2 Summary of each chapter

This document sheds new light on functions and algorithms on the domain of functions. We briefly sketch the main result of each chapter.

Chapter 1 The first chapter defines the multiscale function as a computational representation of real functions of the continuous domain. Between the discrete function set 共$(\mathbb{R}^m)$ and the real set $\mathbb{R}^n \to \mathbb{R}^m$ there exists some sort of isomorphism that makes it possible to evaluate functional operators in either space.

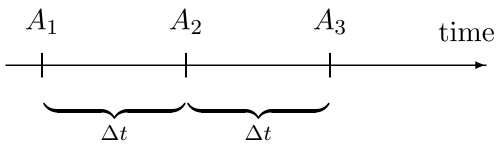
$$\text{共}(\mathbb{R}^m) \;\cong\; \bigcup_{n\in\mathbb{N}} \mathbb{R}^n \to \mathbb{R}^m \tag{0.2}$$

In the context of multiscale calculus a rich set of functional operators are derived. These operators are defined by discrete properties and approximate the results from continuous calculus. A major design goal is also the possibility for straight forward computer implementation.

Chapter 2 The notation of functional operators can be used to describe the quantitative implications of action sequences and trading strategies. Despite the complex definition of some operators we can use them in a plain sequence to describe the flow of activities.

$$\cdots A_1 \Theta^{\Delta t} A_2 \Theta^{\Delta t} A_3 \cdots \tag{0.3}$$

Assume $A_1 \cdots A_3$ to be activities that finish in instantaneous time and $\Theta^{\Delta t}$ to be an operator that waits a time step Δt. The flow of activities is denoted by the chronological order of the operators.



Chapter 3 Chapter three focuses on the time step operator Θ. This operator is part of the explicit operator notation in chapter 2, but was not defined as discrete operator in chapter 1.

$$\Theta^{\Delta t} f(\mathrm{x}) = \mathbb{E}\big(f(X_{t+\Delta t})|X_t = \mathrm{x}\big) \tag{0.4}$$

Obviously, the operator depends on the stochastic process for X. Several common processes for stock prices and interest rates can be expressed in terms of a shift and a blur operator, T and B, that do have a multiscale equivalent.

Chapter 4 The final chapter puts everything into practice and shows how to apply it to a simple problem in quantitative finance. With the operators it is very easy to switch between algebraic and numeric solutions, derive algebraic properties and preprocess numeric evaluations. Also the implementation of adaptivity is just straight forward, once all the operators from chapter 2 are implemented.

0.3 Operator index

This document introduces functional operators for economic effects and for numeric evaluation strategies. The table below briefly indicates the behavior of each operator.

Operator	Interpretation	Chapter
Π	Economic investment strategy	2, 4
Θ	A time step modeled by a stochastic process	2, 3, 4
T	Transfer of goods or assets. Like $\mathcal{T}$ with constant exponent.	1, 2
$\mathcal{T}$	Continuous drift. Solves the transport equation	3, 4
B	Convolution with normal distribution. Like $\mathcal{B}$ with constant exponent	1, 2, 4
$\mathcal{B}$	Random noise added to a variable. Solves the heat equation	3, 4
L	Lévy distributed noise. Solves a convolution	2
$<_{B}^{A}$	Piecewise function. Random events or options	1, 2, 3, 4
S	Invert a function on the unit interval	1
Δ	Discrete differential	1
$\int \cdot \Delta \mathrm{x}$	Discrete integration	1, 4
P	Interpolation of data	1
A	Adaptive interpolation	1, 4
$\boxplus$	Sparse grid interpolation for high dimensions	1, 4
T	Discrete translation operator	1
Z	Discrete dilatation operator	1, 4
大	Evaluation operator, insert values	1, 2, 3, 4

1 Multiscale Calculus

Calculus has been developed as the premier method to denote physical and economic processes. For a long time the analytic approach presented the only viable method to find the solutions of variational problems. As calculations became more complex and computational resources allowed increasingly accurate approximations, the key role of calculus has shifted to preprocessing and outlining numericalal computations. Multiscale calculus provides a discrete extension to calculus such that algorithmic and numerical concepts can be expressed.
The suggested calculus is based on three operators for translation, dilatation and evaluation. It thereby combines the concepts of lambda calculus and multiresolution analysis to achieve a computational representation of mathematical functions on continuous domains. Since the concept can be implemented directly in various programming languages it serves as a general data type for numerical functions upon which functional operators can be implemented. Thus, it provides a unified interface to computer algebra systems and numerical packages.

1.0 Introduction

Calculus is heavily based on the concept of functions and their algebraic transformations. Computational difficulties arise with the numerous mathematical operators that require function evaluations at infinitely many positions. Integration and differentiation are examples. If no closed-form solution can be found, many different techniques exist to estimate the result based on a finite selection of evaluations. This computational inconsistency always appears when limits prevent the formula to reduce to a finite set of operations.
The discretization of continuous functions can be solved by a variety of incompatible algorithmic concepts. Computer algebra systems imitate a mathematician's work with pencil and paper. They can get quite far in deriving and simplifying mathematical operations. Although results can be short and exact, you will easily drop out of the class of representable objects as soon as no closed-form solution is available. The next common representation is an array of discrete function samples. Highly optimized algorithms work on this kind of data structure, since all linear operators are expressed as matrix multiplications. Serious drawbacks of plain arrays are the "curse of dimension" and the lack of adaptivity. More sophisticated techniques involve basis functions such as b-splines or wavelets. These can occur in adaptive resolutions and in arbitrary arrangements in higher dimensions. Although wavelets have ideal algorithmic properties, it is extremely difficult to synthesize complex operators. Another concept of a mathematical function is the function type in programming languages. As they have to be defined entirely in program code, they lack the flexibility to represent arbitrary functions at run time.

1.1 Multiscale functions

A multiscale function is a scale dependent representation of a mathematical function. It is based on three important achievements in computer science: multiresolution analysis, lambda calculus and object oriented programming. Wavelets are a very efficient discretization of mathematical functions on continuous domains. Real valued functions can be expressed by a discrete set of multiresolution coefficients, which are identified by integer translation and dilatation parameters. Lambda calculus provides a computational notation for the function type as used by computer algebra systems. Lambda terms are as flexible as normal data and behave like functions when evaluated. The object oriented programming paradigm was most successful in unifying the interfaces to several data structures and algorithms. One common implementation interface can provide access to all analytic functions.

Set of multiscale functions The set of multiscale functions over the range type V will be written by the symbol $\text{朮}(V)$. We will first analyze some basic properties of the set and its elements and then present the mathematical procedure for transforming mathematical functions into multiscale functions including their representation in program code. The set of multiscale functions $\text{朮}(V)$ will be shown to be in some sense isomorph to the set of functions that map the continuous domain $\mathbb{R}^n$ onto range V. The equivalent function set $\mathbb{R}^n \to V$ spans all finite but arbitrary dimensional functions including extended functions such as the delta function and differential forms.

$$\underbrace{\text{朮}(V)}_{\substack{\text{set of}\\ \text{multiscale}\\ \text{functions}}} \quad \cong \quad \underbrace{\left(\bigcup_{n\in\mathbb{N}} \mathbb{R}^n \to V\right)}_{\substack{\text{set of real}\\ \text{functions}}} \tag{1.1}$$

1.1.1 Axioms

The entire multiscale calculus is based on basic algebra and can in most parts be applied and understood without explicit reference to continuous calculus. In analogy to the definition of vector spaces we define the space of multiscale functions as a five-tuple, that consists of a set of V valued multiscale functions M, an underlying vector space V, the evaluation operator 大 and two vectors of directed translation and dilatation operators T and Z.

$$\text{朮}(V) = (M, V, \text{大}, \mathsf{T}, \mathsf{Z}) \tag{1.2}$$

Just as in vector spaces we define the members of the function space by the membership of the function set M.

$$f \in \text{朮}(V) \;\Leftrightarrow\; f \in M \tag{1.3}$$

The operators 大, T and Z are called evaluation, translation and dilatation operator respectively. The latter may also be refered to as zoom operator.

Their domain is as follows. 大 maps elements from the function space into the underlying vector space V. T and Z are vectors of countable dimensionality with each component an operator, that maps one function on another function, according to shortly defined rules. Each component in the operator vector refers to another direction.

$$\begin{aligned} \text{大} &: \text{𡗗}(V) \rightarrow V \\ \mathsf{T} &: \left(\text{𡗗}(V) \rightarrow \text{𡗗}(V)\right)^{\mathbb{N}} \\ \mathsf{Z} &: \left(\text{𡗗}(V) \rightarrow \text{𡗗}(V)\right)^{\mathbb{N}} \end{aligned} \tag{1.4}$$

For 𡗗(V) to be a valid set of functions it must further fulfill several constraints on the behavior of the operators.

Axiom 1: Scaling condition The scaling condition defines the relationship between applications of a translation operator and a zoom operator acting in the same direction i.

$$\forall i \in \mathbb{N}: \quad \mathsf{Z}_i \, \mathsf{T}_i = \mathsf{T}_i^2 \, \mathsf{Z}_i \tag{1.5}$$

This definition actually implies a dilatation by a factor of two. One step on the coarse resolution is equivalent to two steps on the refined resolution.

Axiom 2: Commutativity All operators T_i and Z_j commute, unless the scaling condition holds.

$$\begin{aligned} \mathsf{T}_i \, \mathsf{T}_j &= \mathsf{T}_j \, \mathsf{T}_i \\ \mathsf{Z}_i \, \mathsf{Z}_j &= \mathsf{Z}_j \, \mathsf{Z}_i \\ \mathsf{T}_i \, \mathsf{Z}_j &= \mathsf{Z}_j \, \mathsf{T}_i \quad \forall i \neq j \end{aligned} \tag{1.6}$$

Given these two rules any operator term consisting of translations and dilatations can be transformed into a **standard form**. All translations, ordered by index are placed left. Dilatations are placed right. The combination operator is then fully determined by the number of translations and the number of zooms in each direction.

Axiom 3: Invertible translation Each translation operator must have an inverse, such that each direction can be traveled back and forth.

$$\forall i \in \mathbb{N}: \quad \exists \mathsf{T}_i^{-1}: \; \mathsf{T}_i^{-1} \mathsf{T}_i = \mathsf{T}_i \mathsf{T}_i^{-1} = \mathit{Id} \tag{1.7}$$

Axiom 4: Convergence A further restriction to multiscale functions is their convergence to real functions. Defined by the operator $\Psi^\star$ the following limit function $\Psi^\star f(\mathrm{x})$ must exist for every x and every dimensionality m.

$$\begin{aligned} \Psi^\star f(\mathrm{x}) &= \lim_{n\to\infty} \text{大} \bigodot_{i=1}^{m} \left(\mathsf{T}_i^{\lfloor 2^n \mathrm{x}_i \rfloor} \mathsf{Z}_i^n \right) f \qquad \forall \mathrm{x} \in \mathbb{R}^m, m \in \mathbb{N} \\ &= \lim_{n\to\infty} \text{大} \left(\mathsf{T}_1^{\lfloor 2^n \mathrm{x}_1 \rfloor} \mathsf{Z}_1^n \right) \left(\mathsf{T}_2^{\lfloor 2^n \mathrm{x}_2 \rfloor} \mathsf{Z}_2^n \right) \cdots \left(\mathsf{T}_m^{\lfloor 2^n \mathrm{x}_m \rfloor} \mathsf{Z}_m^n \right) f \end{aligned} \tag{1.8}$$

Having defined a multiscale function we can convert it into the equivalent real function through well defined application of the multiscale operators. The function $\Psi^\star$ maps the set of multiscale functions onto the set of real functions.

$$\Psi^\star : \text{共}(\mathbb{R}) \to (\mathbb{R}^n \to \mathbb{R}) \tag{1.9}$$

A multiscale function f is turned into the real function $\Psi^\star f$ by limiting the number of zoom operators Z to infinity. After n dilatations the translation requires 2^n steps times the original coordinate x to reach the desired position where the function can be evaluated with 大.

Congruence of real and multiscale functions Two functions will be called congruent if they fall within one equivalence class of the real representation. Real functions are represented in multiscale analysis only by a countable number of coefficients and are thereby confined to the space of square integrable functions $\mathscr{L}^2$ [JS93]. Thus we consider functions as congruent if their representation difference has zero mass with respect to the 2-norm.
Let $f \in \text{共}(V)$ and $g \in (\mathbb{R}^n \to V)$, then

$$f \cong g \;\Leftrightarrow\; \|(\Psi^\star f) - g\|_2 = 0. \tag{1.10}$$

1.1.2 From real functions to multiscale functions

The transformation of a function on the real domain into a multiscale function is performed according to a wavelet decomposition into scaling coefficients. We need to define a function Ψ that maps a real function a multiscale function, inverting the effect of $\Psi^\star$.

$$\Psi : (\mathbb{R}^n \to \mathbb{R}) \to \text{共}(\mathbb{R}) \tag{1.11}$$

The first step is to define the five tuple according to (1.2). The set M of multiscale functions is the set of ϕ-Riemann-integrable real valued functions.

$$M_\phi := \left\{ f \in (\mathbb{R}^n \to \mathbb{R}) \,\middle|\, \int_{\mathbb{R}^n} |f(x)|\phi(x)dx < \infty \right\} \tag{1.12}$$

Access to function values is only granted through the multiscale operators 大, T and Z.

$$\text{共}(\mathbb{R}) = \Big(M_\phi, \mathbb{R}, \text{大}, \mathsf{T}, \mathsf{Z} \Big) \tag{1.13}$$

For the definition of the multiscale operators we need some sort of mother scaling function ϕ, that has unit mass and compact support. We will not rely on other typical properties of scaling functions, but with further restrictions to ϕ standard wavelet decomposition could be obtained.

$$\int_{\mathbb{R}^\mathbb{N}} \phi(\mathrm{x})d\mathrm{x} = 1 \tag{1.14}$$

Based on the mother scaling function we can define the effect of all three operators and thus determine the multiscale function uniquely. The evaluation operator 大 is defined as the value of the inner product of a function f with the mother scaling function ϕ. The translation operator T_i maps the multiscale function f onto a new function that is shifted by one unit in direction i. The dilatation operator Z yields a function that is scaled in direction i, by substracting half of the i-th component.

$$\begin{aligned} \text{大} f &= < f, \phi > \\ \mathsf{T}_i f &= \mathrm{x} \rightarrow f\left(\mathrm{x} + \mathbf{e}_i\right) \\ \mathsf{Z}_i f &= \mathrm{x} \rightarrow f\left(\mathrm{x} - \frac{1}{2}\mathrm{x}_i \mathbf{e}_i\right) \end{aligned} \tag{1.15}$$

Wheras the vector $\mathbf{e}_i$ is the i-th unit vector.

Checking the validity The multiscale function defined by (1.15) is valid, since it does not contradict any of our axioms. The proof for the correctness of the operator definitions is pretty straight forward. The four axioms are one by one shown to hold. First we test the scaling condition.

$$\begin{aligned} \mathsf{Z}_i \mathsf{T}_i f &= \mathsf{Z}_i \left[\mathrm{x} \rightarrow f(x + \mathbf{e}_i)\right] \\ &= \mathrm{x} \rightarrow f(\mathrm{x} + \mathbf{e}_i - 1/2\mathrm{x}_i \mathbf{e}_i) \\ &= \mathrm{x} \rightarrow f(\mathrm{x} + 2\mathbf{e}_i - 1/2\mathbf{e}_i(\mathrm{x}_i + 2)) \\ &= \mathsf{T}_i^2 \left[\mathrm{x} \rightarrow f(\mathrm{x} - 1/2\mathrm{x}_i \mathbf{e}_i)\right] \\ &= \mathsf{T}_i^2 \mathsf{Z}_i f \end{aligned} \tag{1.16}$$

The commutativity rules hold.

$$\mathsf{T}_i \mathsf{T}_j f = \mathrm{x} \rightarrow f(\mathrm{x} + \mathbf{e}_i + \mathbf{e}_j) = \mathsf{T}_j \mathsf{T}_i f \tag{1.17}$$

$$\mathsf{Z}_i \mathsf{Z}_j f = \mathrm{x} \rightarrow f(\mathrm{x} - \tfrac{1}{2}\mathrm{x}_i \mathbf{e}_i - \tfrac{1}{2}\mathrm{x}_j \mathbf{e}_j) = \mathsf{Z}_j \mathsf{Z}_i f \tag{1.18}$$

$$\mathsf{T}_i \mathsf{Z}_j f = \mathrm{x} \rightarrow f(\mathrm{x} + \mathbf{e}_i - \tfrac{1}{2}\mathrm{x}_j \mathbf{e}_j)) = \mathsf{Z}_j \mathsf{T}_i f \tag{1.19}$$

The inverse of the translation operator T_i^{-1} subtracts the vector that was added by T_i. There is no alternative solution for this inversion.

$$\mathsf{T}_i^{-1} f = \mathrm{x} \rightarrow f(\mathrm{x} - \mathbf{e}_i) \tag{1.20}$$

The proofs have so far been independent of the definition of 大. The evaluation operator will develop its crucial role when returning from a multiscale functions back to the equivalent real functions.

Recovering the equivalent real function

We will that the operator Ψ forms a correct isomorphism between real functions and multiscale functions. A real function g must be reobtained after a conversion into a multiscale function Ψg.

Lemma: The term $\Psi^\star\Psi f$ revoveres the orginal function $f \in (\mathbb{R}^n \to V)$, if f is Riemann integrable.

$$\Psi^\star\left(\Psi f\right) \cong f \qquad \text{for } f \in (\mathbb{R}^n \to \mathbb{R}) \tag{1.21}$$

We can write the function $\mathrm{y} \to f(\mathrm{y})$ for Ψf according to (1.15) and use definition (1.8) for $\Psi^\star$.

$$\begin{aligned}
\left(\Psi^\star\Psi f\right)(\mathrm{x}) &= \lim_{n\to\infty} 大 \bigodot_{i=1}^{m} \left(\mathsf{T}_i^{\lfloor 2^n \mathrm{x}_i \rfloor} \mathsf{Z}_i^n\right) \left(\mathrm{y} \to f(\mathrm{y})\right) \\
&= \lim_{n\to\infty} 大 \left(\bigodot_{i=1}^{m} \mathsf{T}_i^{\lfloor 2^n \mathrm{x}_i \rfloor}\right) \left(\mathrm{y} \to f(2^{-n}\mathrm{y})\right) \\
&= \lim_{n\to\infty} \left\langle \mathrm{y} \to f\left(2^{-n}\mathrm{y} + \sum_{i=1}^{m} \frac{\lfloor 2^n \mathrm{x}_i \rfloor}{2^n} \mathbf{e}_i\right), \phi(\mathrm{y}) \right\rangle \\
&\overset{\text{Lebesgue}}{=} \left\langle \lim_{n\to\infty} f\left(2^{-n}\mathrm{y} + \sum_{i=1}^{m} \frac{\lfloor 2^n \mathrm{x}_i \rfloor}{2^n} \mathbf{e}_i\right), \phi(\mathrm{y}) \right\rangle
\end{aligned} \tag{1.22}$$

The result is an inner product of a function f that is stretched infinitely wide by the factor 2^{-n} and shifted by x. If the function f is continuous in x this results in $f(\mathrm{x})$ since ϕ has unit mass and compact support.

$$\overset{\exists f'(\mathrm{x})}{=} \langle f(\mathrm{x}), \phi(\mathrm{y}) \rangle = f(\mathrm{x}) \tag{1.23}$$

If function f is Riemann integrable then there can only be countable many discontinuities and the multiscale function is $\mathscr{L}^2$ equal to $f(\mathrm{x})$. We proved that any ϕ-Riemann-integrable function can be represented as multiscale function. In compliance with our initial claim (1.1) a multiscale function f does not change in the 2-norm after transformation into a real function $\Psi^\star f$.

1.2 Computer implementation

This section shows the computer implementation of a multiscale function in various programming languages. The resulting software truly represents a real function and behaves according to analytic results of calculus. The programing interface to multiscale functions will be called Dadim[1]. The data structure for the multiscale function is a direct translation of the three multiscale operators into algorithmic procedures, that comply to the multiscale axioms and allow the definition of algebraic operations.

1.2.1 Java

In the object oriented world of Java, Dadim is an interface. It offers three methods that can be combined to access function results. The value returned by `da()` can be any object of the underlying range type, typically `Double` or

---

[1] da = Chinese for big, dim = short for dimension

`Boolean`. The `trans` method translates the multiscale function by an integer length into any direction. It works in place, i.e. without instantiating new objects, to avoid time consuming copying of data structures. The `zoom` method returns a dilated copy of the Dadim. It is called less frequently and returns a new object, that can be translated independently from the original Dadim.

```
// Licensed under the Gnu Public License
public interface Dadim {
  public Object da();
  public void trans(int dir, int length);
  public Dadim zoom(int dir);
}
```

A Dadim object is not necessarily `Cloneable`. The preferred way to create equivalent copies of a Dadim is to use a multiplexer (see 1.6.2), that simulates multiple instances of a Dadim by translating to each position as requested.

Constant Possibly the most trivial, but also, the most commonly used Dadim is the constant Dadim. The Java class `DadimConstant` encodes the constant function.

$$\texttt{new DadimConstant(c)} \;\cong\; \mathrm{x} \to c \tag{1.24}$$

The class definition `DadimConstant` implements the three methods for the constant function. Every call to the method `da` returns a constant value. Translations are ignored. Dilatations return a self reference.

```
// constant value Dadim
public class DadimConstant implements Dadim {
  Object value;
  public DadimConstant(Object value) {
    this.value= value;
  }
  public Object da() {
    return value;
  }
  public void trans(int dir, int len) {
  }
  public Dadim zoom(int dir) {
    return this;
  }
}
```

1.2.2 C++

The syntax of the C++ implementation is similar to Java, due to the close relationship of both languages. Dadim in C++ is a class with three abstract methods for evaluation, translation and dilatation. A virtual destructor `~Dadim` has to be defined explicitly as one of the peculiarities of C++. A small but decisive difference between C++ and Java is the availability of template definitions. In C++ we can define the template class `V` as the range of the multiscale function.

```
// Licensed under the Gnu Public License
template<class V>
class Dadim {
public:
  virtual ~Dadim() {};
  virtual V da()                            = 0;
  virtual void trans(int dir, int length) = 0;
  virtual Dadim<V>* zoom(int dir)         = 0;
}
```

Every Dadim in C++ has its value type `V` defined at compile time. The standard Dadim type for numerical applications, called `DDadim`, is a pointer on a double typed Dadim.

```
typdef Dadim<double>* DDadim;
```

Variable Probably the second most important multiscale function, after the constant, is implemented in the `DadimVariable`. This function is proportional to the distance from the origin and thus counts the number of translations in a certain direction. According to our isomorphism (1.8) the C++ class `DadimVariable` encodes a linear function in one component. When the sequence of Z operators is applied according to the definition of $\Psi^\star$ this multiscale function converges to a linear function on the continuous domain.

$$\texttt{new DadimVariable(i)} \cong \mathrm{x} \to \mathrm{x}_i \tag{1.25}$$

The class definition of `DadimVariable` has three member variables, a direction index, the current position and the current resolution. Every translation in the variable direction updates the position according to the length and the resolution. With every dilatation in the variable's direction the resolution is divided by two.

```
class DadimVariable : public Dadim<double> {
  int mydir;
  double pos, res;
 public:
  DadimVariable(int dir, double pos=0.0, double res=1.0)
  : mydir(dir), pos(pos), res(res) {
  }
  double da () {
    return pos;
  }
  void trans(int dir, int length) {
    if (dir == mydir)
      pos += res * length;
```

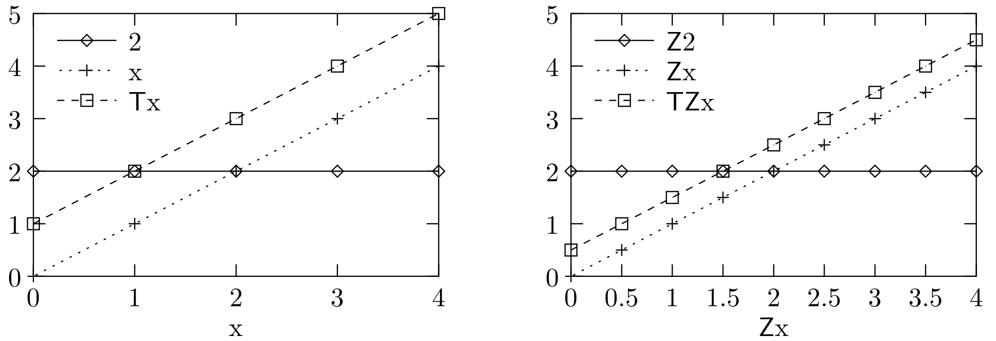


Figure 4: Plotting different multiscale functions against the x variable. On the y-axis there is a constant, a variable and a shifted variable. Operator indices are omitted in this one dimensional example.

```
  }
  DDadim zoom(int dir) {
    return
      new DadimVariable(mydir, pos, dir==mydir? 0.5*res: res);
  }
}
```

In order to extract the function values of a multiscale function we either define the computer program `print` or use the multiscale operators. Both approaches are equivalent.

$$\texttt{print(f, i, N)} \equiv \left(大\mathsf{T}_i^n f\right)_{n=0..N-1} \tag{1.26}$$

The C++ program that prints the sequence of function values alternates evaluations with 大 and translations in direction `index`.

```
void print(DDadim dadim, int index, int N) {
  for (int x=0; x<N; ++x) {
    cout << dadim->da() << endl;
    dadim->trans(index, 1);
  }
}
```

A small table compares the notation in multiscale calculus, C++ and the resulting output. The variables x_i are used synonymously to the function $\mathrm{x} \rightarrow \mathrm{x}_i$.

Calculus	Code	Result
$\left(大\mathsf{T}_0^n\mathrm{x}_0\right)_{n=0..4}$	`print(new DadimVariable(0), 0, 5)`	0, 1, 2, 3, 4
$\left(大\mathsf{T}_1^n\mathrm{x}_2\right)_{n=0..4}$	`print(new DadimVariable(2), 1, 5)`	0, 0, 0, 0, 0
$\left(大\mathsf{T}_0^n\mathsf{Z}_0\mathrm{x}_0\right)_{n=0..6}$	`print(new DadimVariable(0)` `->zoom(0), 0, 7)`	0, 0.5, 1, 1.5, 2, 2.5, 3

Figure 4 visualizes the constant and the linear function graphically.

1.2.3 Maple

Maple is a computer algebra system that supports symbolic and numerical computations. Maple has a powerful programing language that allows procedural and functional styles. The Dadim concept can be expressed in functional and in object oriented, but not in procedural languages. Therefore the functional approach is chosen. Since Maple is an untyped language it is impossible to define a Dadim type. Instead, we define the three multiscale operators for a triplet containing a value, a translation rule and a dilatation rule.

```
# Licensed under the Gnu Public License
da:=(dadim)-> dadim[1];
trans:=(dir, len, dadim)-> dadim[2](dir, len);
zoom:=(dir, dadim)-> dadim[3](dir);
```

Constant The concept of the triplet is best explained with an example function. We define the function `cons` that maps a value onto a constant multiscale function with that value. The first component of the Dadim triplet is simply the value. The second component is a function that maps a translation direction and length onto the translated function, here the same constant. Obviously, this self reference can only be processed by lazy evaluation, since a complete expansion of this expression would run into an infinite loop. The same applies for the dilatation rule, where the only argument is the direction index.

```
cons:=c->[c, (dir, len) -> cons(c), (dir) -> cons(c)];
```

This computer code is the close interpretation of the multiscale operators for the constant $(\mathrm{x} \to c)$. Each entry of the triplet refers to the evaluation rule of one operator.

Code		Calculus
`cons:=c-> [`	`c,`	$\text{大}(\mathrm{x} \to c) \;=\; c$
	`(dir, len) -> cons(c),`	$\mathsf{T}_j^n(\mathrm{x} \to c) \;=\; (\mathrm{x} \to c)$
	`(dir) -> cons(c) ];`	$\mathsf{Z}_j(\mathrm{x} \to c) \;=\; (\mathrm{x} \to c)$

Obviously the constant value c is always returned from the constant function. It is the first component of the Dadim triplet.

```
> da(cons(5));
```

```
5
```

Multiscale functions can have other than numeric range, as this famous example shows.

```
> da(trans(0,1,cons("Hallo World")));
```

```
Hallo World
```

Of coarse we can shift and zoom this function in each direction and always retain the same constant.

Variable We have seen an implementation of the variable as a slightly more exciting example function in the C++ section. Now, we write a functional program for the univariate linear function. For every dimension i we consider the variable x_i. Functions of type $\mathrm{x} \to \mathrm{x}_i$ are inserted for definition of the multiscale operators (1.15) and we get a recursive definition for the multiscale variable x_i. The initial value ($\text{大}\mathrm{x}_i$) of zero could have been chosen differently, but with the application of Z's converges to zero anyway ($\Psi^\star \mathrm{x}_i(0) = 0$).

$$
\begin{aligned}
\text{大}\,\mathrm{x}_i &= 0 \\
\mathsf{T}_j \mathrm{x}_i &= \begin{cases} \mathrm{x}_i + 1 & \text{for } i = j \\ \mathrm{x}_i & \text{for } i \neq j \end{cases} \\
\mathsf{Z}_j \mathrm{x}_i &= \begin{cases} \frac{1}{2}\mathrm{x}_i & \text{for } i = j \\ \mathrm{x}_i & \text{for } i \neq j \end{cases}
\end{aligned}
\tag{1.27}
$$

This recursive definition of the multiscale operators implies the recursive computer code `x(i)` for the variable. The Dadim triplet consists of the value zero, a dilatation rule that adds a constant one per shift length and a dilatation rule that divides the variable by two.

```
x:=(i)->[
0,
(dir,len)->
   piecewise(dir=i,
      x(i) + cons(len),
      x(i)),
(dir)->
   piecewise(dir=i,
      x(i) / 2,
      x(i))
];
```

Some example runs of this code show the concept in practice.

Calculus	Code	Result
$\text{大}\,\mathsf{T}_0\,\mathrm{x}_0$	`da(trans(0,1,x(0)));`	1
$\text{大}\,\mathsf{T}_1^2\,\mathrm{x}_3$	`da(trans(1,2,x(3)));`	0
$\text{大}\,\mathsf{T}_0^3\,\mathsf{Z}_0\,\mathrm{x}_0$	`da(trans(0,3,zoom(0,x(0))));`	3/2

Nested functions A crucial feature of functional calculus is the ability to nest functions, i.e. to insert functional results as arguments of other functions. Assume F to be a function with n arguments and f_1 to f_n multiscale functions.

$$
F \in V^n \to V \wedge f_1, \cdots, f_n \in \text{大}(V) \;\Rightarrow\; F(f_1, \cdots, f_n) \in \text{大}(V) \tag{1.28}
$$

The multiscale functions f_i can be inserted as arguments of function F, whereas the whole term turns into a new multiscale function defined by below multiscale operators.

$$\begin{aligned} 大\ F(f_1,\cdots f_n) &= F(大f_1,\cdots,大f_n) \\ \mathsf{T}_iF(f_1,\cdots f_n) &= F(\mathsf{T}_if_1,\cdots,\mathsf{T}_if_n) \\ \mathsf{Z}_iF(f_1,\cdots f_n) &= F(\mathsf{Z}_if_1,\cdots,\mathsf{Z}_if_n) \end{aligned} \tag{1.29}$$

This function nesting can be implemented by a Maple function `nested` that takes a list of arguments `arg` and a function F.

```
nested:=(arg,F)->[
  F(map(da, arg)),
  (dir,len)->nested(map(a->trans(dir, len,a), arg),F),
  (dir)->nested(map(a->zoom(dir,a), arg), F)];
```

With `nested` all explicit functions are representable as computer algorithms. Crucial for more sophisticated functions is the pointwise product of two Dadim functions. We define the functions `mult` and `div` for the basic arithmetic operations.

```
> mult:=(dadim1,dadim2)->nested([dadim1, dadim2], x->x[1]*x[2]);
  div:= (dadim1,dadim2)->nested([dadim1, dadim2], x->x[1]/x[2]);
```

For + and − the equivalent definition is implicitly assumed by Maple.

Example: The multiplication of the constant functions 3 and 4, for instance evaluates to 12. This example might seem trivial but will reappear in advanced formulas.

$大(3 \times 4) = 大3 \times 大4 = 12$

```
> da(mult(cons(3),cons(4)));
```

```
                                   12
```

1.2.4 Other languages

The Dadim Api can be implemented in all object oriented and all functional languages. Functional languages have their main advantage in the briefness of the code and the similarity to mathematical calculus. For efficient implementation the OOP approach is preferable.

Haskell Haskell is a striktly typed functional language. The data type Dadim is defined over an underlying range type `a`. The three multiscale operators `da`, `trans` and `zoom` work on the the Dadim triple consisting of a value, a translation rule and a dilatation rule.

```
-- Licensed under the Gnu Public License
data Dadim a = DadimTriple (a)
                           (Int -> Int -> Dadim a)
                           (Int -> Dadim a)
```

```
da :: Dadim a -> a
da (DadimTriple e t z) = e

trans :: Int -> Int -> Dadim a -> Dadim a
trans d l (DadimTriple e t z) = t d l

zoom :: Int -> Dadim a -> Dadim a
zoom d (DadimTriple e t z) = z d
```

The obligatory constant function is implemented as function `cons` and maps value of type `a` into a multiscale function over type `a`.

```
cons :: a -> Dadim a
cons c = DadimTriple c (\dir len -> cons c) (\dir -> cons c)
```

In the usual procedure the constant hello world function can be generated.

```
> da (cons "Hallo World")
```

“Hallo World”

Lisp Lisp is an untyped functional language with long standing history. The Dadim type is implicitly defined through the three multiscale operators. A valid Dadim triple consists of a value, a rule for translation and a rule for dilatation. Corresponding components are extracted by each operator and then supplied with the required arguments.

```
;; Licensed under the Gnu Public License
(defun da (dadim) (nth 0 dadim))
(defun trans (dir length dadim)
             (apply (nth 1 dadim) '(dir length)))
(defun zoom (dir dadim) (apply (nth 2 dadim) '(dir)))
```

1.3 Basic calculus

The foundation of calculus is the concept of two complementary operations. The differentiation computes the slope of a function. The integration computes a function from its slope. Both, calculus and multiscale calculus are based on the differential and integral operation to express the variety of mathematical and physical concepts.

1.3.1 Differential

The differential operator Δ measures the change of a multiscale function when moved by one unit into the differentiated direction. Since the resolution of the translation operator is halved with every dilatation, the operator Δ converges towards zero, when applied to a continuous function.

For the definition of multiscale functions it is sufficient to define the results of the three multiscale operators, since they provide the only method to access the function's values. Thus a function is uniquely defined by a triplet containing the value, a translation rule and a dilatation rule. For operators, i.e. functions of multiscale functions, we need to specify how the three multiscale operators are evaluated after the functional operator (here Δ_i) is applied.

$$\begin{aligned} \text{大}\,\Delta_i &= \text{大}\mathsf{T}_i - \text{大} \\ \mathsf{T}_j\Delta_i &= \Delta_i\mathsf{T}_j \\ \mathsf{Z}_j\Delta_i &= \Delta_i\mathsf{Z}_j \end{aligned} \tag{1.30}$$

The implementation of the differential operator Δ_i in a functional language is straight forward. The function `Delta` takes two arguments, a direction index `i` and the operand `dadim`.

```
Delta:=(i,dadim) -> [
  da(trans(i,1,dadim)) - da(dadim),
  (dir, len) -> Delta(i,trans(dir,len,dadim)),
  (dir)      -> Delta(i,zoom(dir,dadim))
];
```

Derivative

The classical derivative is computed as the ratio of two differentials. We will use the following notation to resemble the discrete form of the derivative operator. Some indices are omitted to increase readability.

$$\frac{\Delta}{\Delta \mathrm{x}_i} f := \frac{\Delta_i f}{\Delta_i \mathrm{x}_i} \tag{1.31}$$

Figure 5 shows an example of a parabola's first and second derivative. The slope needs some zooms to converge to its analytic result of $2\mathrm{x} - 2$.

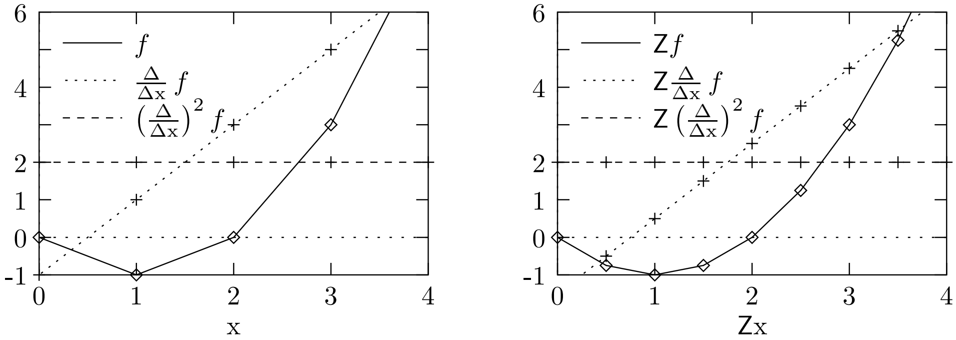


Figure 5: First and second derivative of the parabola $f = \mathrm{x}^2 - 2\mathrm{x}$. With every application of the zoom operator the discrete derivative converges towards the real result.

Lemma: The optical similarity of the real and the discrete differential operators are justified by an $\mathscr{L}^2$ equivalence of both operations.

$$\frac{\Delta}{\Delta \mathrm{x}_i} \cong \frac{d}{d\mathrm{x}_i} \tag{1.32}$$

Proof: The correctness of the differential operator is verified by applying it to a multiscale function f. The result is turned into a real function with $\Psi^\star$ and evaluated at x, which is equal to the analytical derivative of $\Psi^\star f$, provided the later exists.

$$\begin{aligned}
\Psi^\star\left(\frac{\Delta}{\Delta\mathrm{x}} f\right)(\mathrm{x}) &= \lim_{n\to\infty} \text{大}\mathsf{T}^{\lfloor 2^n \mathrm{x}\rfloor}\mathsf{Z}^n \frac{\Delta f}{\Delta \mathrm{x}} && (1.33)\\
&= \lim_{n\to\infty} \frac{\text{大}\Delta\mathsf{T}^{\lfloor 2^n \mathrm{x}\rfloor}\mathsf{Z}^n f}{\text{大}\Delta\mathsf{T}^{\lfloor 2^n \mathrm{x}\rfloor}\mathsf{Z}^n \mathrm{x}}\\
&= \lim_{n\to\infty} \frac{\text{大}\Psi^\star f(\mathrm{y} \mapsto 2^{-n}(\mathrm{y}+1)+\mathrm{x}) - \Psi^\star f(\mathrm{y}\mapsto 2^{-n}\mathrm{y}+\mathrm{x})}{2^{-n}}\\
&= \lim_{n\to\infty} \frac{\Psi^\star f(2^{-n}+\mathrm{x}) - \Psi^\star f(\mathrm{x})}{2^{-n}}\\
&= \frac{d}{d\mathrm{x}}\Psi^\star f(\mathrm{x})
\end{aligned}$$

Derivative rules

A decisive advantage of real over discrete calculus is the existence of nicer derivative rules. Whereas discrete calculus always produces terms that can be evaluated numerically. One rule that holds in real and in multiscale calculus is the linearity of the derivative. Equation (1.35) shows that the product rule for the discrete case has just one additional term.

Lemma: Let $f, g \in \text{共}(V)$

$$\begin{aligned}
\Delta_i(f+g) &= \Delta_i f + \Delta_i g && (1.34)\\
\Delta_i fg &= f\,\Delta_i g + g\,\Delta_i f + (\Delta_i f)\,(\Delta_i g) && (1.35)
\end{aligned}$$

Proof: The proof for the product rule is the only thing that might raise some difficulties. The T and the Z operators act on Δ as if it was the identity operator. So we have to prove the correctness only for the evaluation operator 大.

$$\begin{aligned}
\text{大}\Delta(fg) &= \text{大}\mathsf{T}(fg) - \text{大}(fg) && (1.36)\\
&= \text{大}(\mathsf{T}f)(\mathsf{T}g) - \text{大}(fg)\\
&= \text{大}(f\mathsf{T}g - fg) + \text{大}(g\mathsf{T}f - fg) + \text{大}(\mathsf{T}f - f)(\mathsf{T}g - g)\\
&= \text{大}g\Delta f + \text{大}f\Delta g + \text{大}(\Delta f)(\Delta g)
\end{aligned}$$

The detailed procedure of proofs in multiscale calculus based on complete induction and extensionality will be explained in section 1.3.3.

Delta function

The delta function is an important function in physics and applied mathematics. It is defined mathematically as a Dirac sequence and has the indicator function $1_{x>0}$ as integral. Since we deal exclusively with function sequences in multiscale calculus, the delta function can be directly written as a derivative.

$$\delta(x) = \frac{\Delta}{\Delta x} 1_{x>0} \tag{1.37}$$

For the implementation of the indicator function we rely on the definition for nested functions (1.29) and define a piecewise function that selects between two values based on a test function that is either greater or lower than zero. The function `ifgz` is slightly more general than needed at this point, but we will reuse this function later.

$$\texttt{ifgz(f0, f1, f2)} \cong \begin{tikzcd}[row sep=small] & f_1 \\ {} \arrow[ur, no head, "f_0>0"] \arrow[dr, no head, "f_0\leq 0"'] & \\ & f_2 \end{tikzcd} \tag{1.38}$$

```
ifgz:= (test, dadim1, dadim2) ->
nested([test,dadim1,dadim2],
       x -> if x[1]>0 then x[2] else x[3] end if);
```

Now we can implement the delta function as the quotient of differentials according to (1.37).

```
> delta:= i-> div(Delta(i,ifgz(x(i),cons(1),cons(0))),
                  Delta(i,x(i)));
```

The delta function does not converge to one specific value. As we increase the computational precision in figure 6 the peak gets higher and thinner. Its mass always stays one.

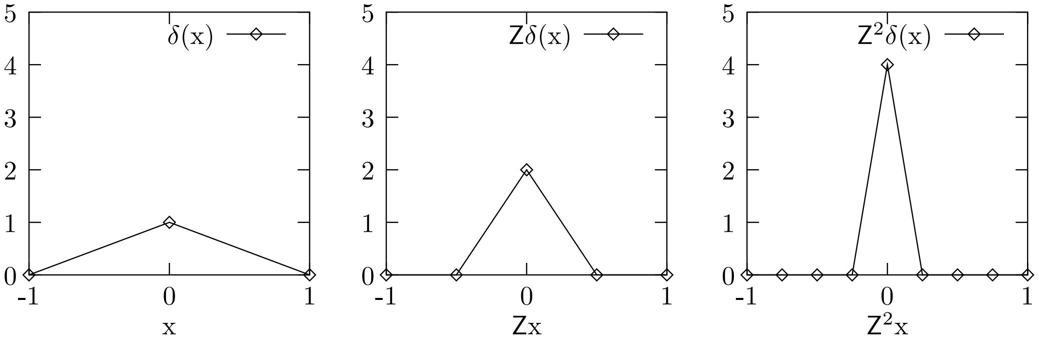


Figure 6: The delta function is the derivative of a discontinuous step function. It is represented in multiscale calculus as a discrete Dirac sequence.

1.3.2 Integral

The integral reverts the effect of the differentiation and computes the area bordered by a function. We will derive the integral operator in two steps. First, we only compute the integral of a function over a unit interval, also known as Haar scaling coefficient. Then we add multiple unit length integrals to gain an integral over an arbitrary interval.

Haar integration

The operator H is defined by a numerical scheme that is borrowed from the wavelet transformation. An application of the dilatation operator refines the transformed function and performs a decomposition into scaling coefficients on the next coarser level. After one application the Z operator yields a sum over two points. With multiple dilatations, the operator H turns into a sumation of all points on a finite interval.

$$\begin{aligned} \text{大}\, H_i &= \text{大} \\ \mathsf{T}_j H_i &= H_i \mathsf{T}_j && \forall i,j \\ \mathsf{Z}_j H_i &= \begin{cases} H_i \mathsf{Z}_i + H_i \mathsf{T}_i \mathsf{Z}_i & \text{for } i = j \\ H_i \mathsf{Z}_j & \text{for } i \neq j \end{cases} \end{aligned} \tag{1.39}$$

The operator does not normalize the mass of the integral and can only be applied to something that is proportional to the resolution $\Delta\mathrm{x}$. The computer code for the Haar integration is straight forward.

```
haar:=(i,dadim)->[
  da(dadim),
  (dir, len) ->
    haar(i, trans(dir,len, dadim)),
  (dir) ->
    piecewise(dir=i,
      haar(i,zoom(dir,dadim) + trans(i,1, zoom(dir, dadim,dir))),
      haar(i,zoom(dir,dadim)))
];
```

Lemma: The H operator computes the integral over the unit interval. When applied to a multiscale function f, it changes the sequence generated by succesive applications of the dilatation operator Z, such that the new sequence converges to $\Psi^\star H f$ which is the integral over $\Psi^\star f$. This concept is expressed in continuous calculus, after the introduction of a temporary variable x'_i and the specification of the integration border.

$$H_i \left(f \Delta \mathrm{x}_i \right) \;\cong\; \int_{\mathrm{x}_i}^{\mathrm{x}_i+1} \Psi^\star f\big|_{\mathrm{x}_i=\mathrm{x}'_i}\, d\mathrm{x}'_i \tag{1.40}$$

Proof: This proof is again focused on the univariate case. Thus, all the indices to T and Z can be omitted. The transformation of a multiscale function into a function on continuous domain $\Psi^\star$ is defined via a limitation to an infinite application of Z (1.8). With each dilatation the sum is refined and converges to the integral over $\Psi^\star f$.

$$\begin{aligned}
\Psi^\star H f \Delta \mathrm{x} &= \lim_{n\to\infty} \text{太}\mathsf{T}^{\lfloor 2^n \mathrm{x}\rfloor}\mathsf{Z}^n H(f\Delta \mathrm{x}) \\
&\overset{(1.39)}{=} \lim_{n\to\infty} \text{太}\mathsf{T}^{\lfloor 2^n \mathrm{x}\rfloor}\mathsf{Z}^{n-1}\left(H\mathsf{Z}(f\Delta \mathrm{x}) + H\mathsf{T}\mathsf{Z}(f\Delta \mathrm{x})\right) \\
&\;\;\vdots \\
&\overset{(1.5)}{=} \lim_{n\to\infty} \text{太}\mathsf{T}^{\lfloor 2^n \mathrm{x}\rfloor}\left(H\mathsf{Z}^n f + H\mathsf{T}\mathsf{Z}^n f + \cdots + H\mathsf{T}^{2^n-1}\mathsf{Z}^n f\right)\Delta \mathrm{x} \\
&\overset{(1.39)}{=} \lim_{n\to\infty} \text{太}\sum_{i=0}^{2^n-1} \mathsf{T}^{\lfloor 2^n \mathrm{x}\rfloor + i}\mathsf{Z}^n(f\Delta \mathrm{x}) \\
&= \lim_{n\to\infty} \sum_{i=0}^{2^n-1} \frac{1}{2^n}\Psi^\star f(\mathrm{x} + \frac{i}{2^n}) \\
&= \int_{\mathrm{x}}^{\mathrm{x}+1} \Psi^\star f(\mathrm{x}')d\mathrm{x}'
\end{aligned} \tag{1.41}$$

Integral operator

The discrete integral operator $\int$ can now be defined as a sum of Haar integrations. The integral starts with the initial value 0 and is increased by the area over a unit interval (H_i), when translated by one unit with T_i.

$$\begin{aligned}
\text{太}\int_i &= 0 \\
\mathsf{T}_j\int_i &= \begin{cases} H_i + \int_i & \text{for } i = j \\ \int_i \mathsf{T}_j & \text{for } i \neq j \end{cases} \\
\mathsf{Z}_j\int_i &= \int_i \mathsf{Z}_j \qquad \forall i, j
\end{aligned} \tag{1.42}$$

Lemma: In compliance with axiom three, the T operator has an uniquely defined inverse for integral terms.

$$\mathsf{T}^{-1}\int = \int - \mathsf{T}^{-1}H \tag{1.43}$$

Proof: What makes the integral definition more complex than previous operators is the non trivial verification of this axiom. The interesting case only occurs when the direction indices of the T and $\int$ match. The proof therefore has univariate form.

$$\begin{aligned}
\mathsf{T}^{-1}\mathsf{T}\int &= \int \quad \Leftrightarrow \\
\mathsf{T}^{-1}\left(\int + H\right) &= \int \quad \Leftrightarrow \\
\mathsf{T}^{-1}\int + \mathsf{T}^{-1}H &= \int \quad \Leftrightarrow \\
\mathsf{T}^{-1}\int &= \int - \mathsf{T}^{-1}H
\end{aligned} \tag{1.44}$$

Implementation The implementation of the integral operator needs to distinguish between forward and backward shifts. Either the translation rule for T^{+1} (1.42) is applied or the derived rule for negative translation with length -1 (1.43). Furthermore, it is not possible to compute the effect of multiple applications of Ts in one step. The result of $\mathsf{T}^n \int$ must be evaluated by n times iterated application of T.

```
integ:=(i,dadim)->[
  0,
  (dir, len)->
     piecewise(dir=i,
       piecewise(
         len>0, trans(dir,len-1,
                      integ(i,dadim) + haar(i,dadim)),
         len<0, trans(dir, len+1,
                      integ(i,dadim) -
                      trans(dir,-1, haar(i,dadim))),
         len=0, integ(i,dadim)),
       integ(i,trans(dir,len,dadim))),
  (dir) ->
     integ(i,zoom(dir,dadim))
];
```

The definite integral

We will use the following notation to resemble the discrete form of the continuous integral. The integrand function f is multiplied with the resolution $\Delta\mathrm{x}$ when integrated with the $\int$ operator.

$$\int f\Delta\mathrm{x}_i := \int_i (f\Delta_i\mathrm{x}_i) \tag{1.45}$$

The definite integral over a certain interval is computed by first shifting the function f to the beginning of the interval, then applying the $\int$ operator and finally shifting everything to the interval's end.

$$\int_a^b f\,\Delta\mathrm{x}_i := \mathsf{T}_i^{b-a} \int (\mathsf{T}_i^a f)\,\Delta\mathrm{x}_i \tag{1.46}$$

Lemma: The relationship between the discrete and the continuous definite integral is as expected, with the only difference, that the integration variable (here x_i) is bounded outside the integral.

$$\int_a^b f\Delta\mathrm{x}_i \cong \int_a^{\mathrm{x}_i+b} f|_{\mathrm{x}_i=\mathrm{x}_i'}\,d\mathrm{x}_i' \tag{1.47}$$

However, if the variable x_i is not used or, in particular, not shifted with a T_i operator, then x_i valuates to zero, according to (1.27).

Proof: The proof relies on the congruency of discrete and continuous integral, as it will be shown in (1.55), $\int f \Delta \mathrm{x}_i \cong (\int_0^{\mathrm{x}_i} f|_{\mathrm{x}_i=\mathrm{x}'_i} d\mathrm{x}'_i)$.

$$\begin{aligned}
\int_a^b (\Psi f)\Delta \mathrm{x}_i &= \mathsf{T}_i^{b-a} \int \mathsf{T}_i^{a}(\mathrm{x} \to f(\mathrm{x}))\Delta \mathrm{x}_i \qquad (1.48)\\
&= \mathsf{T}_i^{b-a} \int (\mathrm{x} \to f(\mathrm{x} + a\mathbf{e}_i))\, \Delta \mathrm{x}_i\\
&\overset{(1.55)}{\cong} \mathsf{T}_i^{b-a} \left(\mathrm{x} \to \int_0^{\mathrm{x}_i} f(\mathrm{x} + a\mathbf{e}_i)\big|_{\mathrm{x}_i=\mathrm{x}'_i} d\mathrm{x}'_i\right)\\
&= \mathrm{x} \to \int_0^{x_i+b-a} f(\mathrm{x} + a\mathbf{e}_i)\big|_{\mathrm{x}_i=\mathrm{x}'_i} d\mathrm{x}'_i\\
&= \mathrm{x} \to \int_a^{x_i+b} f(\mathrm{x})\big|_{\mathrm{x}_i=\mathrm{x}'_i} d\mathrm{x}'_i
\end{aligned}$$

1.3.3 Fundamental theorem of multiscale calculus

The fundamental theorem of calculus states the inverse relationship between integral and differential. In multiscale calculus this property is even more important than good approximations to the continuous result. Derived mathematical concepts rely on the exactness of the inversion rather than any other property of the integral.

Lemma: This first lemma confirms a previously postulated congruence of H and the integral (1.40).

$$\Delta_i H_i = \mathsf{T}_i - \mathit{Id} \qquad (1.49)$$

Proof: By the principle of extensionality we can prove the correctness of our lemma for all feasible evaluation paths. Since access to multiscale function is only possible through the three operators, the following combination of 大, T and Z represent all function values. Again, we are only interested in the univariate problem. All involved operators behave like identity operators in all other directions.

$$\forall n \in \mathbb{Z}, m \in \mathbb{N}_0 : \quad 大\,\mathsf{T}^n\,\mathsf{Z}^m \Delta H = 大\,\mathsf{T}^n\,\mathsf{Z}^m(\mathsf{T} - \mathit{Id}) \qquad (1.50)$$

The complete induction over the parameters n and m starts with $n = m = 0$ and performs induction steps over n and m. The translation length n can be negative. Thus the induction step for n must be performed in a positive and a negative direction.

$$
\begin{array}{llll}
\text{大}\,\Delta H & & & \\
 & = & \text{大}H\mathsf{T} - \text{大}H & \\
 & \overset{(1.39)}{=} & \text{大}\,\mathsf{T}H - \text{大}\,H & \\
 & = & \text{大}\,\mathsf{T} - \text{大} & = \text{大}\,(\mathsf{T} - \mathit{Id}) \\
\text{大}\,\mathsf{T}^{n\pm 1}\Delta H & \overset{(1.30)}{=} & \text{大}\,\mathsf{T}^{n}\Delta \mathsf{T}^{\pm 1}H & \\
 & \overset{(1.39)}{=} & \text{大}\,\mathsf{T}^{n}\Delta H\mathsf{T}^{\pm 1} & \\
 & = & \text{大}\,\mathsf{T}^{n}(\mathsf{T} - \mathit{Id})\mathsf{T}^{\pm 1} & = \text{大}\,\mathsf{T}^{n\pm 1}(\mathsf{T} - \mathit{Id}) \\
\text{大}\,\mathsf{T}^{n}Z^{m+1}\Delta H & \overset{(1.30)}{=} & \text{大}\,\mathsf{T}^{n}\mathsf{Z}^{m}\Delta \mathsf{Z}H & \\
 & \overset{(1.39)}{=} & \text{大}\,\mathsf{T}^{n}\mathsf{Z}^{m}\Delta(H\mathsf{Z} + H\mathsf{T}\mathsf{Z}) & \\
 & = & \text{大}\,\mathsf{T}^{n}\mathsf{Z}^{m}(\Delta H\mathsf{Z} + \Delta H\mathsf{T}\mathsf{Z}) & \\
 & = & \text{大}\,\mathsf{T}^{n}\mathsf{Z}^{m}(\mathsf{T}\mathsf{Z} - \mathsf{Z} + \mathsf{T}^{2}\mathsf{Z} - \mathsf{T}\mathsf{Z}) & \\
 & \overset{(1.5)}{=} & \text{大}\,\mathsf{T}^{n}\mathsf{Z}^{m}(\mathsf{Z}\mathsf{T} - \mathsf{Z}) & = \text{大}\,\mathsf{T}^{n}\mathsf{Z}^{m+1}(\mathsf{T} - \mathit{Id})
\end{array}
\tag{1.51}
$$

Theorem: The fundamental theorem of multiscale calculus states the inversity of $\int$ and Δ. Applying the differential operator Δ to the integral operator $\int$, both with matching direction indices, yields the identity.

$$\Delta_i \int_i = \mathit{Id} \tag{1.52}$$

Proof: The theorem is verified in univariate calculus for all translation and dilatation paths.

$$\forall n \in \mathbb{Z}, m \in \mathbb{N}_0 : \quad \text{大}\,\mathsf{T}^{n}\,\mathsf{Z}^{m}\Delta \int = \text{大}\,\mathsf{T}^{n}\,\mathsf{Z}^{m}\mathit{Id} \tag{1.53}$$

The complete induction starts with $n = m = 0$ and requires two induction steps. The first proves the theorem's correctness for all non zero translations and the second for all dilatations.

$$
\begin{aligned}
\text{大}\Delta\textstyle\int &= \text{大}\mathsf{T}\textstyle\int - \text{大}\textstyle\int \\
&\overset{(1.42)}{=} \text{大}(H + \textstyle\int) - 0 \\
&= \text{大}H + \text{大}\textstyle\int \qquad = \text{大} \\
\text{大}\mathsf{T}^{n+1}\Delta\textstyle\int &\overset{(1.30)}{=} \mathsf{T}^{n}\Delta\mathsf{T}\textstyle\int \\
&= \text{大}\mathsf{T}^{n}\Delta(\textstyle\int + H) \\
&= \text{大}\mathsf{T}^{n}(\Delta\textstyle\int + \Delta H) \\
&\overset{(1.50)}{=} \text{大}\mathsf{T}^{n}(\mathsf{T} - \mathit{Id} + \mathit{Id}) \qquad = \text{大}\mathsf{T}^{n+1} \\
\text{大}\mathsf{T}^{n-1}\Delta\textstyle\int &\overset{(1.30)}{=} \mathsf{T}^{n}\Delta\mathsf{T}^{-1}\textstyle\int \\
&\overset{(1.43)}{=} \text{大}\mathsf{T}^{n}\Delta(\textstyle\int - \mathsf{T}^{-1}H) \\
&= \text{大}\mathsf{T}^{n}(\Delta\textstyle\int - \mathsf{T}^{-1}\Delta H) \\
&\overset{(1.50)}{=} \text{大}\mathsf{T}^{n}(\mathit{Id} + \mathsf{T}^{-1} - \mathit{Id}) \qquad = \text{大}\mathsf{T}^{n-1} \\
\text{大}\mathsf{T}^{n}\mathsf{Z}^{m+1}\Delta\textstyle\int &\overset{(1.30)}{=} \text{大}\mathsf{T}^{n}\mathsf{Z}^{m}\Delta\mathsf{Z}\textstyle\int \\
&\overset{(1.42)}{=} \text{大}\mathsf{T}^{n}\mathsf{Z}^{m}\Delta\textstyle\int\mathsf{Z} \qquad = \text{大}\mathsf{T}^{n}\mathsf{Z}^{m+1}
\end{aligned}
\tag{1.54}
$$

Lemma: Now, the congruency between the discrete and the continuous integration will be shown. If we apply the discrete integral operator our result is the integral function of the operand f with zero as the lower integration border.

$$
\int (\Psi f)\Delta \mathrm{x}_i \cong \int_0^{\mathrm{x}_i} f|_{\mathrm{x}_i=\mathrm{x}_i'}\, d\mathrm{x}_i' \tag{1.55}
$$

Proof: The first part of the proof shows that the integral operator creates an integral function, based on its inversity to the discrete differential and the congruency of discrete and continuous differentials.

$$
\begin{aligned}
\Psi^{\star} f &\overset{(1.52)}{=} \Psi^{\star}\frac{\Delta}{\Delta \mathrm{x}_i}\int f\Delta \mathrm{x}_i \\
&= \frac{d}{d\mathrm{x}_i}\Psi^{\star}\int f\Delta \mathrm{x}_i \\
\int (\Psi^{\star} f) d\mathrm{x}_i &= \Psi^{\star}\int f\Delta \mathrm{x}_i
\end{aligned}
\tag{1.56}
$$

The final part of this proof searches for the correct integration border. Given that the integral function is always zero at the origin, due to (1.42) and (1.8), zero must be the starting point of the integration.

$$
\left(\Psi^{\star}\int f\Delta \mathrm{x}_i\right)(0) \quad = \quad 0 \qquad \forall f \tag{1.57}
$$

1.4 Advanced calculus

Solving advanced problems in multiscale calculus is already very close to computer programing. Apart from finding a solution and presenting it nicely, we might be concerned with convergence properties and efficiency issues. Due to the equidistant and regular sampling of multiscale functions it is possible to employ a wide range of existing algorithms. We will investigate the algorithm of nested intervals and some techniques known from finite differences. Furthermore we will convert the general convolution integral into discrete calculus and discuss its representation for optimal convergence.

1.4.1 Solver

The discrete samples of a multiscale function are ideally suited for many numerical algorithms. The solver operator S implements the algorithm of nested intervals to find a function's zero point. For some argement function f, the first approximation of the zero point is based on a linear interpolation of $f(0)$ and $f(1)$. When dilated, the zero point is recursively searched in one of the subintervals. The presented code is limited in its applicability, but is a powerful tool when used correctly. The operand to S must be monotonously increasing and have its zero point between 0 and 1.

$$\begin{array}{rcl} \text{大}\, S_i f &=& -\dfrac{\text{大} f}{\text{大}\,\Delta_i f} \\ \mathsf{T}_j S_i &=& S_i \mathsf{T}_j \\ \mathsf{Z}_i S_i &=& \begin{cases} \frac{1}{2} S_i \mathsf{Z}_i & \mathsf{T}_i \mathsf{Z}_i > 0 \\ \frac{1}{2} + \frac{1}{2} S_i \mathsf{T}_i \mathsf{Z}_i & \end{cases} \\ \mathsf{Z}_j S_i &=& S_i \mathsf{Z}_j \qquad \text{for } i \neq j \end{array} \tag{1.58}$$

The code is still simple and has infinite room for amendments. Simplicity, on the other hand, can be an advantage when complexity and convergence have to be analyzed precisely. The implementation uses the if-greater-zero function `ifgz` from (1.38).

```
solver:= (i, dadim) -> [
  -da(dadim)/(da(Delta(i,dadim)))),
  (dir,len) ->
    solver(i, trans(dir,len,dadim)),
  (dir) ->
    piecewise(dir=i,
      ifgz(trans(i,1,zoom(i,dadim)),
           solver(i,zoom(i,dadim))/2,
           cons(1/2) + solver(i,trans(i,1,zoom(i,dadim)))/2),
      solver(i,zoom(dir,dadim)))
];
```

Example: On zoom level two the algorithm of nested intervals S computes the square root of 1/2 with an error of 0.0071.

$\mathsf{Z}^2\, S(\mathrm{x} \mapsto \mathrm{x}^2 - \frac{1}{2}) = 0.7$

```
> da(zoom(0,zoom(0,solve(0,mult(x(0),x(0))-cons(1/2)))));
```

$$\frac{7}{10}$$

The solver operator can be used in combination with the derivative to find local optima. Section 2.4.2 makes use of the maximization to find optimal investment parameters within a trading strategy.

Lemma: For a continuous and monotoneously increasing function $g \in (\mathbb{R}^n \to \mathbb{R})$ with $g(0) < 0$ and $g(1) > 0$:

$$g(\Psi^{\star} S \Psi g) = 0. \tag{1.59}$$

Proof: We define a sequence g_i of real valued functions with

$$g_i = \Psi^{\star} \mathsf{Z}^i S \Psi g. \tag{1.60}$$

For $i = 0$ the term g_0 is obviously the zeropoint of the line through g at coordinate 0 and 1, according to (1.58).

$$\text{大} S \Psi g_0 = -\frac{g_0(0)}{g_0(1) - g_0(0)} \tag{1.61}$$

For later elements of the sequence the zero point is either searched in the left subinteval $[0, 1/2]$ or in the right $[1/2, 1]$ depending on the evaluation of $g(1/2) = \text{大}\mathsf{T}\mathsf{Z}\Psi g$. The selected interval is transformed onto the unit interval $[0, 1]$ and the search algorithm can be called recursively. The returned zero point is a relative coordinate on that domain and is trasformed back by multiplying $1/2$ and adding $1/2$ in case of $g(1/2) < 0$.

$$\Psi g_{i+1} = \begin{cases} \frac{1}{2} S \Psi g_i(\frac{1}{2}\mathrm{x}) & \text{for } g(1/2) > 0 \\ \frac{1}{2} + \frac{1}{2} S \Psi g_i(\frac{1}{2}\mathrm{x} + \frac{1}{2}) & \text{otherwise} \end{cases} \tag{1.62}$$

1.4.2 Blur operator

The blur operator plays an important role in statistics and physics. Physically it solves the heat equation over the unit time interval and models the dispersion of energy under diffusion. Statistically, the operator performs a convolution of a function with the Gaussian density. Thus we can compute the density of a random variable after adding normally distributed noise. In section 3.2.2 we will see this operator as a main ingredient for the synthesis of stochastic models for economic parameters. The operator B_i performs the convolution in direction i. In calculus it is easier to define the operator for the variance up to 1/2.

$$B_i := B_i^{\frac{1}{2}} B_i^{\frac{1}{2}} \tag{1.63}$$

The definition of B is justified by the central limit theorem. On the coarsest level the operator 大 evaluates a discretely sampled expected value of a random variable with variance $\sigma \leq 1/2$. When the resolution is refined the operator has to be applied four times as often.

$$\begin{aligned}
\text{大}\ B_i^\sigma &= \text{大}\frac{\sigma}{2}\mathsf{T}_i^{-1} + (1-\sigma)\text{大} + \frac{\sigma}{2}\text{大}\mathsf{T}_i^1 \qquad (1.64)\\
\mathsf{T}_j B_i^\sigma &= B_i^\sigma \mathsf{T}_j\\
\mathsf{Z}_j B_i^\sigma &= \begin{cases} \left(B_i^\sigma\right)^4 \mathsf{Z}_i & \text{for } i=j\\ B_i^\sigma \mathsf{Z}_j & \text{for } i \neq j \end{cases}
\end{aligned}$$

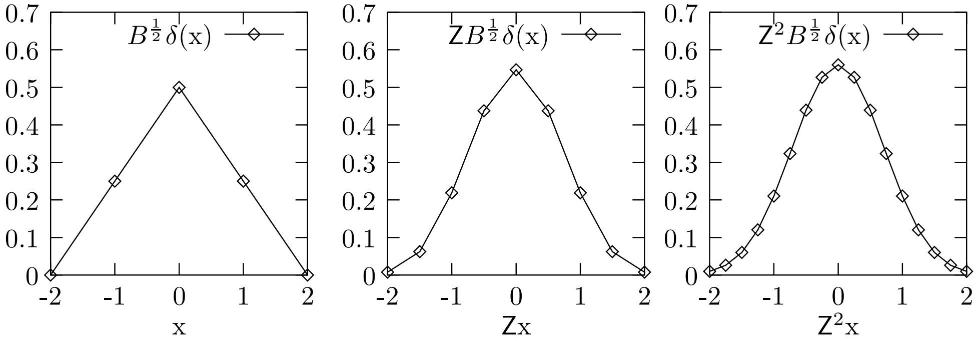


Figure 7: The Gaussian bell curve is generated by an application of the blur operator to the delta function.

Lemma: The application of the B operator is equivalent to a convolution with the normal distribution.

$$B_i f \cong \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{u^2}{2}} \Psi^\star f|_{\mathrm{x}_i=\mathrm{x}_i+u}\, du \qquad (1.65)$$

Proof: Consider x_i to be a stochastic variable that performs a random jump. It jumps up to $\mathsf{T}_i\mathsf{Z}_i^n\mathrm{x}_i = \mathsf{Z}_i^n\mathrm{x}_i + 1/2^n$ with probability $\sigma/2$, stays where it is with $1-\sigma$ and drops to $\mathsf{T}_i^{-1}\mathsf{Z}_i^n\mathrm{x}_i = \mathsf{Z}_i^n\mathrm{x}_i - 1/2^n$ with a chance of $\sigma/2$.

$$\begin{aligned}
\mathrm{Var} &= \sigma/2 \times (-1/2^n)^2 + (1-\sigma)\times 0 + \sigma/2 \times (1/2^n)^2 \;=\; 2^{-2n}\sigma \qquad (1.66)\\
\mathrm{Mean} &= \sigma/2 \times (-1/2^n) + (1-\sigma)\times 0 + \sigma/2 \times (1/2^n) \;=\; 0
\end{aligned}$$

Apparently, on zoom level n the operator B^σ performs a convolution with probability density with mean 0 and variance $2^{-2n}\sigma$. According to the definition of $\mathsf{Z}B^\sigma$, this convolution is repeated 4^n times.

$$\begin{aligned}
\mathrm{Var}(\mathsf{Z}^n B^\sigma) &= 4^n \mathrm{Var}(B^\sigma) = \sigma \qquad (1.67)\\
\mathrm{Mean}(\mathsf{Z}^n B^\sigma) &= 4^n \mathrm{Mean}(B^\sigma) = 0
\end{aligned}$$

Following the central limit theorem, the convolution kernel can only be a normal distribution with variance σ and mean 0.

1.4.3 Continuous shift operator

For the solution of the transport equation, also applied in economics for the transfer of goods, we define a continuous operator T. Section 2.4.1 will use this operator as a possible activity in a trading strategy. Section 3.2.1 applies the shift operator to introduce deterministic effects to economic parameters. The T operator is equivalent to the multiscale operator T, but has an extended domain for its exponent. The exponent v is an extra argument of multiscale type $v \in \mathbb{大}(\mathbb{R})$.

$$\begin{aligned}
大 T_i^v &= 大(1 - v + \lfloor v \rfloor)\, \mathsf{T}_i^{\lfloor 大 v \rfloor} + 大(v - \lfloor v \rfloor)\, \mathsf{T}_i^{\lceil 大 v \rceil} \qquad (1.68)\\
\mathsf{T}_j T_i^v &= T_i^{\mathsf{T}_j v} \mathsf{T}_j \\
\mathsf{Z}_j T_i^v &= \begin{cases} T_i^{2\mathsf{Z}_i v} \mathsf{Z}_i & \text{for } i = j \\ T_i^{\mathsf{Z}_j v} \mathsf{Z}_j & \text{for } i \neq j \end{cases}
\end{aligned}$$

Lemma: The continuous shift operator T_i^v shifts the argument function f by v units into direction i. Alternatively we can interpret this operator as an increase of variable x_i by v.

$$T_i^v f \cong \mathrm{x} \mapsto \Psi^\star f(\mathrm{x} + v\mathbf{e}_i) \qquad (1.69)$$

Proof: The proof is quite simple when focusing on the i direction, where v is considered constant. The T and the Z operators are defined according to the multiscale axioms and 大 evaluates a linear interpolation if 大v is not of integer type.
Let $g \in \mathbb{R}^n \to V$. We can compute the limit of $T^v \Psi g$.

$$\begin{aligned}
(\Psi^\star T^v \Psi g)(\mathrm{x}) &= \lim_{n\to\infty} 大\mathsf{T}^{\lfloor 2^n \mathrm{x} \rfloor} \mathsf{Z}^n T^v \Psi g \qquad (1.70)\\
&\overset{(1.68)}{=} \lim_{n\to\infty} 大\mathsf{T}^{\lfloor 2^n \mathrm{x} \rfloor} T^{2^n v} (\mathrm{y} \mapsto g(2^{-n}\mathrm{y}) \\
&\ \ \text{let } p \in \mathbb{N},\ q \in [0,1[\text{ and } 2^n v = p + q \\
&\overset{(1.68)}{=} \lim_{n\to\infty} 大 T^{p+q} (\mathrm{x} \mapsto g\big(2^{-n}(\mathrm{y} + \lfloor 2^n \mathrm{x} \rfloor)\big) \\
&\overset{(1.68)}{=} \lim_{n\to\infty} (1-q) g(2^{-n}(\lfloor 2^n \mathrm{x} \rfloor + p)) + q g(2^{-n}(\lfloor 2^n \mathrm{x} \rfloor + p + 1)) \\
&= \lim_{n\to\infty} f(\mathrm{x} + 2^{-n} p) \ = \ f(\mathrm{x} + v)
\end{aligned}$$

Multiple examples for this operator will be given in section 2.4.1, where this operator is put into practice.

Conservation of mass The T operator is often used in physics and economics to model a transport of energy or matter. Without special care the T operator changes the functions integral value. In order to avoid this issue we can apply a sequence of operators. First we integrate the function, then transport the mass regardless of any losses and finally differentiate the result.

$$\widetilde{T}_i^v \ := \ \Delta_i\, T_i^v \int_i \qquad (1.71)$$

The transformed transport operator $\widetilde{T}$ can conserve the operand's mass precisely.

$$\textstyle\int_i \widetilde{T}_i^v = \int_i \Delta_i T_i^v \int_i = T_i^v \int_i \tag{1.72}$$

1.4.4 Convolution

The convolution is a binary operation that combines two functions. Commonly it is used in statistics to compute the density or the distribution of a random variable, after a random noise was added. Section 3.1.1 uses convolutions to express a stochastic model for economic parameters in its most general form. We assume $\Phi \in \mathcal{C}^1$ to be the noise's probability distribution, monotone, smooth and with unit mass, and f to be any ϕ-integrable function. The usual notation of the distribution is transformed onto the unit interval, where it can be converted into discrete calculus.

$$\begin{aligned} \int_{-\infty}^{\infty} \Phi'(u) f(\mathrm{x}-u) du &= \int_0^1 f\left(\mathrm{x} + \Phi^{-1}(u)\right) du \qquad (1.73) \\ &\cong \int_0^1 T_{\mathrm{x}}^{\Phi^{-1}(u)} f \Delta u \end{aligned}$$

Example: Figure 8 shows the convolution of the delta function with the Poisson density, for which the inverse distribution Φ^{-1} can be given analytically.

$$\Phi^{-1}(u) = -\ln(1-u) \tag{1.74}$$

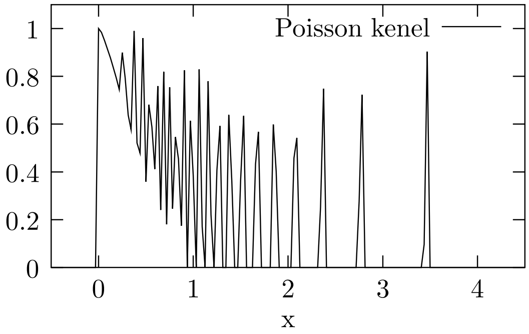


Figure 8: Convolution of the Poisson density with a delta function on zoom level 5. The mass is one and the center of mass converges. The absolute value, however, varies between zero and one.

Higher order In the infinitesimal world, there is an exact equivalence of the convolution integral and its partially integrated version.

$$\begin{aligned} \int_0^1 f(\mathrm{x}+\Phi^{-1}(u)) du &= \int_0^1 \frac{\frac{d}{du} \int_0^{\mathrm{x}+\Phi^{-1}(u)} f(\mathrm{x}') d\mathrm{x}'}{\frac{d}{du} \Phi^{-1}(u)} du \qquad (1.75) \\ &\cong \int_0^1 \frac{\Delta_u T_{\mathrm{x}}^{\Phi^{-1}(u)} \int_{\mathrm{x}} f}{\Delta_u \Phi^{-1}(u)} \Delta u \end{aligned}$$

This equivalence does not hold in the discrete world, but expresses the same mathematical idea and converges to the same limit function. Which of the two versions is to be used can be chosen freely and decided upon preferred convergence properties.

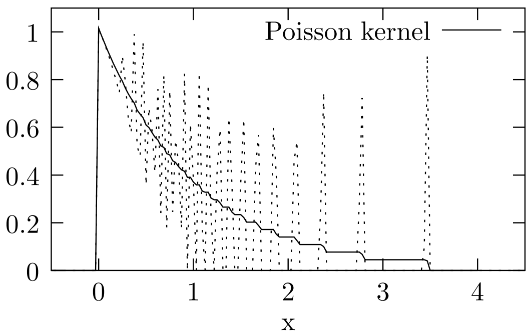


Figure 9:
Higher order representation of the convolution. Where the delta spikes were before, there are now only steps.

1.5 Adaptivity

This section will derive a key argument for the implementation of the Dadim API in numerical applications. One of the most effective acceleration techniques in computational algorithms is adaptivity. Through sophisticated strategies the computational effort can be focused on the problem's most relevant parts, while smooth areas are accurately evaluated on low resolutions. However, adaptive algorithms usually require elaborate data structures and, when implemented, still need some extra tuning in either theoretical consideration or experimental optimization. Wouldn't it be nice to have adaptive strategies that can be applied and tested for a multiscale function just as easily as any differential operator? This section will derive three adaptivity operators that exactly lead to this kind of adaptive evaluations.

1.5.1 Interpolation

Before we can turn to the definition of adaptive strategies we need to define an interpolation method. Suppose a user decides to perform a dilatation to increase the computational precision. Normally, this command is passed to the whole multiscale function. What can be done to protect smooth and already accurate parts from expensive dilatation? The answer is to pretend increased accuracy by interpolating previous results. In calculus this means to write the interpolation operator W instead of the dilatation operator Z. When and how this is performed is part of the adaptive strategy.

The interpolated function is constructed by the two dual operators W and $\overline{W}$. The first operator W evaluates the function on grid points that also existed on the coarse grid, while its dual $\overline{W}$ evaluates the new positions. A reasonable guess for the values on the refined grid at coincident points are the corresponding values on the coarse grid. The interpolation operator W turns into its dual operator $\overline{W}$ when translated.

$$\begin{aligned} \text{大}\, W_i &= \text{大} \\ \mathsf{T}_j W_i &= \begin{cases} \overline{W}_i & \text{for } i = j \\ W_i \mathsf{T}_j & \text{for } i \neq j \end{cases} \\ \mathsf{Z}_j W_i &= W_i \mathsf{Z}_j \end{aligned} \tag{1.76}$$

For the intermediate grid points we can interpolate their values by a linear combination of surrounding coarse values. If not specified otherwise we can assume a piecewise linear scheme ($h_0 = h_1 = 1/2$). When translated or dilated, the operator $\overline{W}$ for intermediate positions turns back into its dual W for coincident positions.

$$\begin{aligned} \text{太}\,\overline{W}_i &= \sum_{k=-K}^{K} h_k \text{太} T_i^k, \qquad \text{with} \sum_{k=-K}^{K} h_k = 1 \\ \mathsf{T}_j \overline{W}_i &= \begin{cases} W_i \mathsf{T}_i & \text{for } i = j \\ \overline{W}_i \mathsf{T}_j & \text{for } i \neq j \end{cases} \\ \mathsf{Z}_j \overline{W}_i &= \begin{cases} W_i \mathsf{Z}_i & \text{for } i = j \\ \overline{W}_i \mathsf{Z}_j & \text{for } i \neq j \end{cases} \end{aligned} \tag{1.77}$$

Example The effect of the intpolation W is illustrated by the following examples. If the interpolated function is translated by an even number, the function is evaluated with half the number of translations. A dilatation of the operand f is simulated. For uneven translations, the intermediate point is interpolated from surrounding evaluations.

$$\begin{aligned} \text{太}\mathsf{T}^{2n}\mathsf{Z}^m W f &= \text{太}\mathsf{T}^n \mathsf{Z}^m f \\ \text{太}\mathsf{T}^{2n+1}\mathsf{Z}^m W f &= \sum_{k=-K}^{K} h_k \text{太}\mathsf{T}^{n+k}\mathsf{Z}^m f \end{aligned} \tag{1.78}$$

1.5.2 A priori adaptivity

The first adaptive strategy in our series is introduced by the predict operator P. Suppose we wanted to add or otherwise combine two functions, one rough and one smooth. Furthermore we know by either a priori knowledge or preceding experiments that for good overall precision it is sufficient to apply m fewer Z's to the smooth function. Thus we can apply m times the P operator to the smooth function. Each P replaces one Z with an interpolation W.

$$\begin{aligned} \text{太}\, P_i &= \text{太} \\ \mathsf{T}_j P_i &= P_i \mathsf{T}_j \\ \mathsf{Z}_j P_i &= \begin{cases} W_i & \text{for } i = j \\ P_i \mathsf{Z}_j & \text{for } i \neq j \end{cases} \end{aligned} \tag{1.79}$$

Examples: When we apply n dilatation operators to m predict operators with matching direction indices we have to distinguish two cases. If n is larger than m the predict operators are completely dissolved into the according number of interpolations. Otherwise, the predict operators survive in a number reduced by n.

$$\mathsf{Z}_i^n P_i^m = \begin{cases} W_i^m \, \mathsf{Z}_i^{n-m} & \text{for } n > m \\ W_i^n & \text{for } n = m \\ W_i^n \, P_i^{m-n} & \text{for } n \leq m \end{cases} \tag{1.80}$$

A special case of prediction is created by an application with infinite power. This is not an adaptivity operator since it converges to a different function when increasing accuracy. However it serves as an important basis for advanced adaptive strategies.

$$\mathsf{Z}_i^n \, P_i^\infty = W_i^n \, P_i^\infty \tag{1.81}$$

1.5.3 A posteriori adaptivity

The next version of our adaptive strategy must have two improvements. First, the decision on where to refine and where to interpolate should be found automatically and, second, the refinement decision should adapt to local function properties. The a posteriori adaptive strategy is executed by the A operator that takes control over a function's smooth and rough areas.

The adaptivity decision is based on the error estimator E that determines, whether an interpolation can generate the desired accuracy or whether the decision must be repeated on the refined level. The decision operator is in fact a function with three arguments, the choice indicator $E < \varepsilon$ and the two alternatives $P^\infty\mathsf{Z}$ and $A\mathsf{Z}$. This form of adaptivity obviously works only if lazy evaluation is assumed. The dilatations of the multiscale function do not change with the application of A. Only the fact where and if at all the dilated function is evaluated does.

$$\begin{aligned} \text{大}\, A_i &= \text{大} \\ \mathsf{T}_j A_i &= A_i \mathsf{T}_j \end{aligned} \tag{1.82}$$

$$\begin{tikzcd} & & P_i^\infty \mathsf{Z}_i \\ \mathsf{Z}_i A_i \arrow[r, phantom, "="] & {} \arrow[ur, dash, "\mathrm{E}\leq\varepsilon"] \arrow[dr, dash, "E>\varepsilon"'] & \\ & & A_i \mathsf{Z}_i \end{tikzcd}$$

$$\mathsf{Z}_j A_i = A_i \mathsf{Z}_j \quad \text{for } i \neq j$$

We can implement a simple error estimator by comparing the effect of refinement with the interpolation $|\mathsf{Z} - W|$. This difference is interpolated on all refined levels with P^∞. The is no need to refine the estimator unless the function itself is. A second term counts the number of refinements and increases the estimator with a factor of α. This ensures that the function still converges correctly when precision is increased.

$$E = P_i^\infty |\mathsf{Z}_i - W_i| - \alpha \log_2(\Delta \mathrm{x}_i) \tag{1.83}$$

Whereas $\text{大}\alpha\mathsf{Z}^n \log_2(\Delta\mathrm{x}_i)$ equals αn. With a large number of zooms, the error estimator always reaches the tolerance ε. In other words, even if the interpolation is perfect, every α/ε-th zoom is executed.

Example: Figure 10 shows a kind of worst case for the adaptive strategy. The function $\sqrt[5]{B^{1/2}\delta(\mathrm{x})}$ is zero at first and converges to its real value only after several Z's are applied.

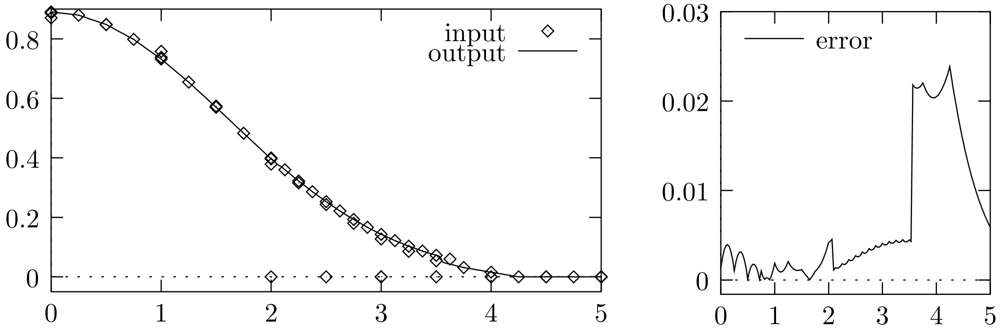


Figure 10: A posteriori adaptive strategy created by $A\sqrt[5]{B^{1/2}\delta(\mathrm{x})}$. The input points on the zero axis show that the function evaluates to zero even after refinement. The plot was computed on level 5 with $\varepsilon = 0.05$ and $\alpha = 0.25\varepsilon$.

1.5.4 Sparse grid

Our final strategy for selective evaluation is the sparse grid. The method presents an efficient and accurate adaptive strategy that works for smooth and high dimensional functions [BD03, Bun98, JGT01]. The rationale behind sparse grids is summarized as follows. Suppose we wanted to evaluate a d-dimensional function f on zoom level l for each direction. On standard grids, the number of Z operators is $d \times l$, each doubling the number of points in the grid. The resulting computational effort of $O(2^{dl})$ exceeds available resources even for relatively low d and l. Here is where the sparse grid enters the scene. Instead of applying all Z's to f at once, the sparse scheme combines the result from multiple versions of f, each with its own selection of zoomed directions. The maximum number of Z's applied in sequence is now l and the total computational effort reduces to $O(l\log(l)^d)$[Bun98].

The sparse grid operator $\boxplus$ is always applied to a set of multiple directions D. Its definition exploits a recursive construction property. Each grid can be combined recursively from two grids, one with lower dimension and one with lower resolution.

$$\begin{aligned}
\text{大} \underset{D}{\boxplus} &= \text{大} \\
\mathsf{T}_i \underset{D}{\boxplus} &= \underset{D}{\boxplus} \mathsf{T}_i \\
\mathsf{Z}_i \underset{D}{\boxplus} &= \begin{cases} \boxplus_{D\setminus\{i\}} P_i^{\infty} W_i + P_{D\setminus\{i\}} \boxplus_D (\mathsf{Z}_i - W_i) & \text{for } i \in D \\ \boxplus_D \mathsf{Z}_i & \text{otherwise} \end{cases}
\end{aligned} \tag{1.84}$$

The predict operator P with multiple indices D refers to a successive application of one P operator for each index direction.

$$P_D = P_{d_1} P_{d_2} \cdots P_{d_n} \qquad \text{for } D = \{d_1, d_2, \cdots, d_n\} \tag{1.85}$$

Example This final example evaluates a discrete derivative on an adaptive sparse grid. The function of interest f is a discrete derivative of an exponential function.

$$f := \frac{\Delta^2}{\Delta \mathrm{x}_1 \Delta \mathrm{x}_2} e^{-(\mathrm{x}_1+2\mathrm{x}_2)} \tag{1.86}$$

For the sparse and adaptive approximation $\widetilde{f}$ to the two dimensional function f three operators are deployed. The sparse grid for both dimensions, and one adaptivity operator for each direction.

$$\widetilde{f} = \underset{\{1,2\}}{\boxplus} A_1 A_2 f \tag{1.87}$$

The result of the adaptive algorithm is quite impressive, given the low amount of code and the ease of use. More advanced adaptive schemes can be developed at the expense of brief and precise formulas.

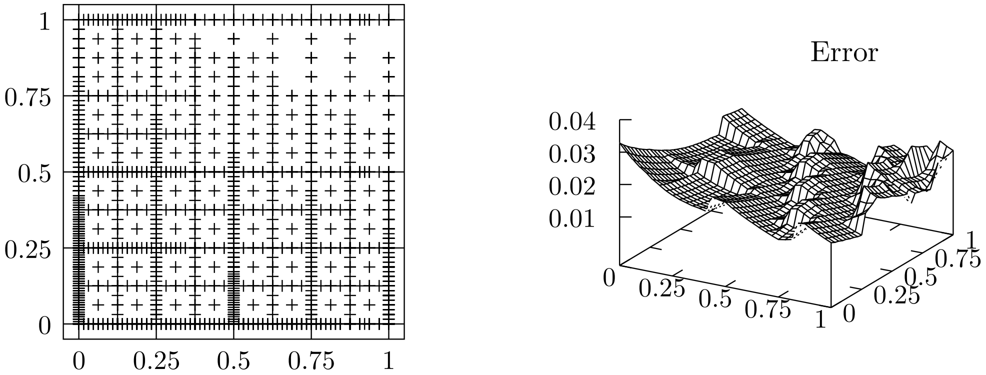


Figure 11: Adaptive sparse grid approximation generated by the mixed strategy $\boxplus_{\{1,2\}} A_1 A_2$. Function f is evaluated adaptively on level eight with an error tolerance of $\varepsilon = 0.01$. Instead of $2^8 \times 2^8 = 64$k calls to f, the result is combined from only 2800 evaluations and is nearly as accurate as a standard grid on level 7.

1.6 Proxy operators

Finally we have to spend a paragraph on practical issues. The implementation of multiscale calculus in a functional language was shown to be short and compact, but without further tuning even simple programs run for ages and do not stop filling giga bytes of main memory. Relief is brought, how else could it be, by extra operators that reorganize the computational procedure. Interestingly, these operators rely only on the Dadim API for multiscale functions. We will consider two kinds of proxy operators that behave like identity operators mathematically, but perform computational optimizations in terms of time and memory requirements. Proxies just work like and were inspired by web proxies in their way to reduce function calls to subsequent units.

1.6.1 Cache

Caches are well known in computer science. They can store the results of previous calculations and reproduce them when the same calculation is about to be repeated. Sounds simple, but isn't necessarily. Dadim caches have to meet several challenges. First, all dilatation and translation paths have to be tracked in order to identify equal evaluation positions. Second, the dimensionality might possibly be unknown at compile time. Many mathematical concepts, like convolutions, introduce temporary dimensions at run time. Third, appropriate heuristics are needed to determine obsolete cache contents. And finally, cache operators must be extremely fast, since we want to install too many of them rather than missing one at an important place.

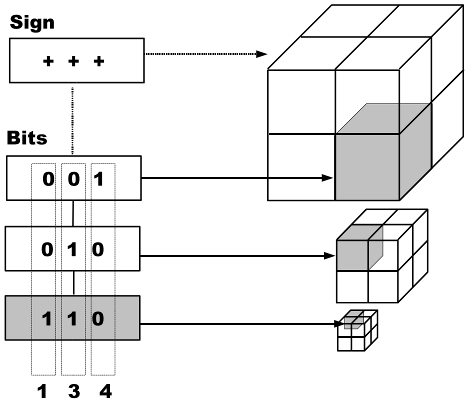


Figure 12: Possible data structure for the Dadim cache. The nested hyper cubes guarantee that nearby values are stored together. Each cube is referenced by a bit set according the n-th bit in each coordinate. For the translation, the lowest bits are first and, in case of an overflow, the next higher bits are then updated

Figure 12 shows a possible data structure for the cache. The plotted tree structure must be implemented twice. One tracks translation paths and contains values and the other one tracks dilatation paths and contains caches of the first kind. It has been shown that optimal memory access pattern follows the vertex order of a space filling curve, e.g. the Hilbert or the Peano curve [GMPZ04].

1.6.2 Multiplexer

The importance of the second proxy operator is easily underestimated. The representation of Dadim objects occupies a significant part of main memory. Many operators reference multiple instances of multiscale functions and exponentially increase this number when dilated. Luckily, these instances mostly differ only in their translation history and can be converted into each other. The Dadim multiplexer simulates multiple instances of a Dadim by tracking the translation history of each virtual instance and executing the translation in case of an evaluation. When dilated, the virtual instance must attach to a new multiplexer, that can be found in a cache for equivalent dilatation paths.

1.6.3 Other proxies

The evaluation procedure of Dadim programs can be tuned further with more advanced proxy routines. Since there is always a trade off between algorithmic overhead and possible accelerations, one must be increasingly careful with more elaborate proxies. While multiplexer and cache must be applied after more or less any operation, the following proxies must be well understood before deployed.

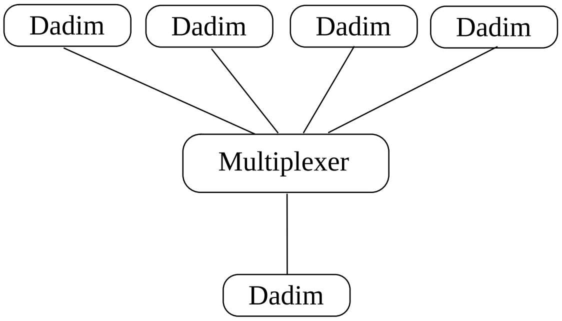


Figure 13: The Dadim multiplexer simulates several instances of a multiscale function. Each time a virtual instance performs an evaluation, the single instance is translated to the virtual's position.

Beylkin sampler The first of our proxies was a cache that could store function values. In a logical step to the next higher complexity we can try to cache the effect of operators. All linear operators perform only linear combinations of the operand's values. Finding the linear weight of each input value is often connected with high computational effort. It is possible to extract these weights by applying the operator to an array of n test functions with a complexity of $O(n^2)$. Once sampled, the operator can be applied to any function by decomposition into these test functions. Represented in an appropriate wavelet basis the operator can be applied with a complexity of $O(n)$. The method was first described by G. Beylkin in [Bey92, Bey91] and pays for all linear operators that are applied extremely often to different functions.

Algebraic simplificator Depending on the computational representation of Dadim objects it is possible to simplify a functional term algebraically. This requires the knowledge of algebraic operator properties like linearity and directional dependencies. The triggers and the simplification rules should be chosen with care, since simplification commands can take a very long time. A suggested strategy is the simplification after each zoom. This command is not executed very often and generates some simplification opportunity in most instances, especially when common operator sequences like $\int \cdots \Delta\mathrm{x}$ or $B\,B$ have a tuned implementation.

Tracer The tracing proxies do not change, but visualize the computational procedure. They store intermediate results, evaluated points and required times into a file for further inspection by the user. Due to the highly recursive character of the programs it is next to impossible to trace the computation in a conventional debugger. The precise action of the adaptivity schemes in the previous section could only be plotted after the application of such a proxy.

1.7 Conclusion

This chapter presented a unified interface to computer algebra systems and numerical packages. It enables the user to specify mathematical terms in a notation close to traditional calculus, embed numeric algorithms in functional and object oriented programming styles, smoothly integrate discrete data samples and express cutting edge adaptive multidimensional algorithms. The core of the discrete calculus is the multiscale function which is a computer implementable algebraic data structure and constitutes a computational interface to functions on the multidimensional real domain. It can be transformed and used with the operations known from continuous calculus. At the same time it induces a unique and well-defined algorithmic evaluation procedure on regular grids of function samples.

2 Pattern in portfolios

This chapter considers sequences of trading activities and suggests a mathematical representation for typical pattern in human investment decisions. In particular we will express financial strategies as sequences of operators that can be implemented according to the formalism introduced in chapter 1. All kinds of financial contracts, strategies and multiperiod games can be captured in terms of quantitative implications by a vocabulary of three words: waiting, transacting and deciding. Each activity will be represented by a functional operator, that can be interpreted in an operator sequence as a chronologically ordered list of instructions. In an example setting we will investigate the behavior of a typical risk averse agent and derive hedging possibilities and the resulting economic impact.

2.0 Introduction

The field of quantitative finance has to deal with an increasing complexity and the rapid development of new contract types. The creation of a consistent and unified portfolio framework is complicated by the lack of a mathematical notation for human trading activities and financial products. Elaborate contracts and investment objectives are mostly specified in prose form and typically use specific terminology that is hard to interpret by an uninvolved. Such representations are difficult to evaluate mathematically and have to be translated into formulas and computer code individually. Those who feel inclined to pursuit greater generality are mostly struck by an inflation of parameters and mathematical concepts.

In order to meet the industry demand for a technical portfolio representation there have been developments in the extension of existing programming dialects. Most notable results are *MLFi* [JES00], based on the functional programing language Caml, and the XML-standard *Fpml* [fpm04]. However, neither provides a mathematical framework for the derivation of theoretical properties. Their vocabulary is huge and gets still extended. Finally, there is a large distance from product representation to the evaluation procedure, which raises the fear of model ambiguity and inconsistency.

Here we suggest an explicit and mathematically precise operator notation for financial portfolios and financial derivatives. The notation is based on the foundations of operator theory and introduces a vocabulary of three operators: waiting, transacting and deciding. Each of the possible activities is represented by an operator, which is written in a chronological list to express a sequence of trading activities. The notation thus provides explicit expressions for contract details and trading strategies including embedded options, minimum guaranties, event triggers and non-delta hedges. Furthermore the notation yields a rather explicit procedure to extract its statistical properties, which can be performed by a computer algebra system or by a numerical scheme that is directly derived from the operator sequence. Chapter 3 will give more datails on how the operators can be translated to those defined in the previous chapter.

2.1 The economic state

The prevailing state of economy at time t is summarized by a vector $X(t)$, that contains all relevant and available knowledge. This includes observable market parameters and private book keeping variables. For the scope of this document we will focus on a number of parameters that will occur in the following examples. As relevant information $X(t)$ we consider the current time t, the stock price S, the short rate r, our wealth in cash c and the number of stocks h in our deposit.

$$X(t) = \big[t, S(t), r(t), c(t), h(t), \cdots\big] \in \mathbb{R}^n \tag{2.1}$$

Other parameters that might occur as economic state are forward rates, exchange rates, stochastic volatilities, moving averages and default probabilities, if not constant. In short, every variable that varies in time must be represented as a component of X.

Initial values We can provide the initial values to our models by replacing every instance of X with $X(0)$. This step is always the first operator in an operator sequence and written by the operator 大.

$$大 V := V|_{X=X(0)} \tag{2.2}$$

As we will see, the operator is required to maintain a chronological operator ordering, in which initial events are written left and final actions on the right hand side.

2.2 Valuation function

A valuation function V is an interpretation of state X. It determines the kind of information we want to extract from a portfolio. The next section will show how to compute the expected value of such a function, after some random events occurred. V might evaluate theoretical contract prices, risk measures or probabilities for certain events to happen after an initial state.

$$V : \mathbb{R}^n \to \mathbb{R}^m \tag{2.3}$$

We will briefly discuss the three most common instances of the valuation function. The first example V_c evaluates the amount on the cash account c. It is applicable whenever we are interested in expected cash profits.

$$V_c\big(X(t)\big) = c(t) \tag{2.4}$$

Additionally, we can take our stock deposit into the balance sheet. The function V_a returns the nominal worth of the stock deposit plus the cash account.

$$V_a\big(X(t)\big) = S(t)h(t) + c(t) \tag{2.5}$$

Finally, we consider a probability measure, that is an indicator function on the cash account greater than x. If we compute the expected value of the indicator function V_p we can determine the probability distribution of our total cash profit.

$$V_p\big(X(t)\big) = 1_{c(t)>\mathrm{x}} \tag{2.6}$$

All risk measures like value at risk and expected shortfall can be derived from the full knowledge of the probability distribution.

2.3 Processes

The time process operator Θ is a function of a function and determines the stochastic model for the state variables. With $\Theta^{\Delta t}$ we can look one step Δt into the future and evaluate the expectation of our function V under the future process state. Whenever the operator occurs in a sequence of event operators it refers to a time step Δt with no activity.

$$\Theta : (\mathbb{R}^n \to \mathbb{R}^m) \to (\mathbb{R}^n \to \mathbb{R}^m) \tag{2.7}$$

The process operator $\Theta^{\Delta t}$ is defined as the expected value of the argument function V applied to tomorrow's state $X(t+\Delta t)$ given a current state $\mathrm{x} = X(t)$. The operator always corresponds to a Markovian process, or one that can be turned into Markovian form.

$$\Theta^{\Delta t} V(\mathrm{x}) := \mathbb{E}\left[V\big(X(t+\Delta t)\big) \middle| X(t) = \mathrm{x} \right] \tag{2.8}$$

Another version of the same operator is given by the probability density $p_{\Delta t}(\mathrm{x},\mathrm{y})$ for the state to travel from x to y within one time step. Explicit formulas for this density are derived below for the most common processes.

$$\Theta^{\Delta t} V(\mathrm{x}) = \int_{\mathbb{R}} p_{\Delta t}(\mathrm{x},\mathrm{y}) V(\mathrm{y}) d\mathrm{y} \tag{2.9}$$

Lemma: Multiple applications of the process $\Theta^{\Delta t}$ evaluate the same expected value as (2.8), but with the new time horizon in the exponent.

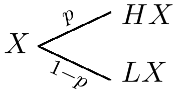
$$\big(\Theta^{\Delta t}\big)^n = \Theta^{n\Delta t} \tag{2.10}$$

Proof: The correctness of the operator power rule (2.10) is verified through repeated application of Θ according to its definition (2.8).

$$\begin{aligned}
(\Theta^{\Delta t})^n V(\mathrm{x}) &= (\Theta^{\Delta t})^{n-1}\, \mathbb{E}\left[V(X(t+\Delta t))\middle|X(t)=\mathrm{x}\right] && (2.11)\\
&= (\Theta^{\Delta t})^{n-2}\, \mathbb{E}\left[\mathbb{E}\left[V(X(t+2\Delta t))\middle|X(t+\Delta t)\right] \middle|X(t)=\mathrm{x}\right]\\
&= (\Theta^{\Delta t})^{n-2}\, \mathbb{E}\left[V(X(t+2\Delta t))\middle|X(t)=\mathrm{x}\right]\\
&\vdots\\
&= \mathbb{E}\left[V(X(t+n\Delta t))\middle|X(t)=\mathrm{x}\right] \qquad = \Theta^{n\Delta t} V(\mathrm{x})
\end{aligned}$$

2.3.1 Discrete process

A simple model but yet a good approximation to reality is the discrete state model. Suppose a random variable X reaches the high state HX with probability p and drops to the low state LX otherwise.



(2.12)

The expected value of a measuring function V is the linear combination of both scenarios. That is just how the expected value was defined in statistics.

$$\Theta V(X) := pV(HX) + (1-p)V(LX) \tag{2.13}$$

Of course we can build multistep trees through multiple applications of Θ. The example below shows the first two steps.

$$\begin{aligned} \Theta^2 V(X) &= \Theta\left[pV(HX) + (1-p)V(LX)\right] \\ &= p\Theta V(HX) + (1-p)\Theta V(LX) \\ &= p^2 V(HHX) + p(1-p)V(LHX) + \\ &\quad\; p(1-p)V(HLX) + (1-p)^2 V(LLX) \end{aligned} \tag{2.14}$$

The result simplifies in recombining trees where the operators L and H commute $LH = HL$.

2.3.2 Brownian motion

The Brownian motion operator $\mathcal{B}_{\mathrm{x}}$ is defined by a convolution with the Gaussian density. It models a jump of the index variable x with a normally distributed jump size. Possible applications will be discussed in section 3.2.2. We will briefly anticipate the result of equation (3.28).

$$\mathcal{B}_{\mathrm{x}}^{\sigma^2} V(\mathrm{x}) = \frac{1}{\sigma\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left(-\frac{1}{2}\left(\frac{\mathrm{x}-\mathrm{y}}{\sigma}\right)^2\right) V(\mathrm{y})d\mathrm{y} \tag{2.15}$$

Example: If the process consisted exclusively of normally distributed variations in variable S with zero drift and standard deviation σ, then we could write:

$$\Theta = \mathcal{B}_S^{\sigma^2}. \tag{2.16}$$

2.3.3 The Black&Scholes model

The Black&Scholes model is a theoretical framework for the dynamics of stock prices and is used for the valuation of stock options [BS73]. Although options are the topic of the next section on financial contracts, we will briefly discuss how the Θ operator is applied in this context. The profit V that we can draw from a call or put option depends on the difference between stock price S and strike price K. In case of a call option we have the right to buy one share at price K. Our profit is consequently $S - K$. The profit is multiplied with the discount factor e^{-rt}.

$$V_{call}(S,t) = \max(S-K,0)e^{-rt} \tag{2.17}$$

In the notation of partial differential equations we write the change of the expected value $\Theta^t V$ in its classical form. The equation yields a unique solution for the expected option value at expiration time.

$$\frac{d}{dt}\Theta^t V = \frac{d}{dt}\Theta^t V + rS\frac{d}{dS}\Theta^t V + \frac{1}{2}\sigma^2 S^2 \frac{d^2}{dS^2}\Theta^t V \tag{2.18}$$

The solution of this equation is well known and can be written explicitly by a convolution with the Gaussian density. The operator Θ_{bs} solves the Black&Scholes formula for the unit time step. The origin of this result will be discussed in more detail in section 3.3.1 and is also found in [Hul02].

$$\Theta_{bs}\, V(S,t) = \int_{\mathbb{R}} \frac{e^{-\mathrm{x}'^2/2}}{\sqrt{2\pi}} V\left(Se^{r+\sigma \mathrm{x}' - \frac{1}{2}\sigma^2}, t+1\right) d\mathrm{x}' \tag{2.19}$$

The expectation of the option payoff at maturity time t is computed by an application of Θ with the appropriate power. The resulting function is supplied with the initial values for S and t.

$$\Theta_{bs}^t V_{call}(S_0, 0) \tag{2.20}$$

Note that the operators are placed in chronological order from left to right. First, you wait t time steps, and then you evaluate the payoff. Although the operators are written left to right, their evaluation direction is opposite. First compute the value function V and then apply Θ_{bs} t times.

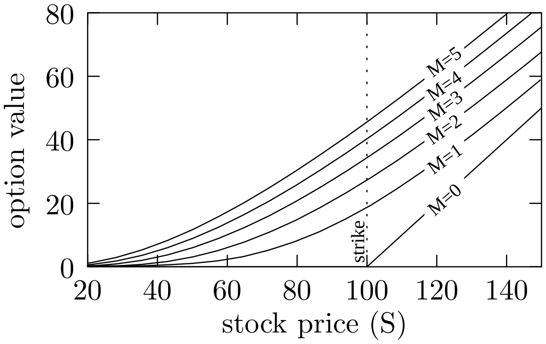


Figure 14: Call option under the Black&Scholes model. The process operator Θ_{bs} is applied t times to V_{call}

Plotted function:
$f_t(S) = {\Theta_{bs}}^t\, V_{call}(S,0)$
with
$\sigma = 40\%,\ r = 5\%,\ K = 100$

The Black&Scholes model describes a stock that increases its value in expectation with the same rate as an interest rate account. The great achievement of Black and Scholes was that option prices can be computed as the expected payoff under the adjusted drift although real stock prices are experienced to grow at a significantly higher rate. Only these option prices can be reproduced without risk in a continuous buying and selling strategy called delta hedge.

2.4 The Portfolio

The ultimate goal of this chapter and the presented operator notation is the ability to describe complex portfolios and contract conditions in a mathematically precise way. We wanted to derive a mathematical term that fully represents our portfolio, including all outstanding transactions, all embedded options, the sensitivity to random events and the room for further trading activity.
A portfolio Π is a function of an operator. Given an operator Θ this results in the new portfolio operator $\Pi(\Theta)$. The effect of the transformation Π is simply an expansion of the considered state variables. While Θ only evaluates the expectation of observable market parameters without any interaction, the portfolio $\Pi(\Theta)$ expands the statistic measure to balance and accounting variables that correspond to the portfolio or the trading strategy.

$$\Pi : ((\mathbb{R}^n \to \mathbb{R}^m) \to (\mathbb{R}^n \to \mathbb{R}^m)) \to ((\mathbb{R}^n \to \mathbb{R}^m) \to (\mathbb{R}^n \to \mathbb{R}^m)) \tag{2.21}$$

A portfolio is specified by a vocabulary of three words: waiting, transacting and deciding. A time step without activity is denoted by Θ. For the transaction we will see the T operator. Decisions will be expressed by a specially designed option operator.

2.4.1 Transaction

The transaction operator T transfers one unit to the accounting variable specified by the operator index. Accordingly, T_{x_i} increases the i-the component of the state vector x by one. We can use the operator power to transfer multiple goods at once.

$$T_{\mathrm{x}_i} V(\mathrm{x}) := V(\mathrm{x} + \mathbf{e}_i) \tag{2.22}$$

Let our value V depend on the amount of units in the account x and some other state variables y. The new value after the delivery of a units is $T_{\mathrm{x}}^{a} V$. The number of transfered goods may vary with the states x and y.

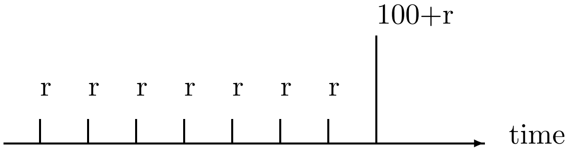
$$T_{\mathrm{x}}^{a(\mathrm{y})} V(\mathrm{x}, \mathrm{y}) = V\left(\mathrm{x} + a(\mathrm{y}), \mathrm{y}\right) \tag{2.23}$$

This operator replaces every occurrence of the index variable with the same variable plus the exponent.

Coupon bond The notation of a coupon bond highlights the use of the operator power to express repeated transactions. The operator term reads chronologically from left to right. First pay a regular coupon rate of size r until maturity M, then pay a redemption of 100.

$$\Pi_c(\Theta) = (\Theta T_c^r)^M T_c^{100} \tag{2.24}$$

The cash flow of a coupon bond is visualized in the plot below.



Swap A swap is an arrangement in which two parties repeatedly exchange assets at predefined conditions. In the case of currency swaps they exchange certain amounts in different currencies over a certain period. For interest rate swaps one pays a fixed size while the other pays a state dependent number. The swap investment Π_s transferes $-r_1$ after half a period and simultaneously transferes $-r_1$ in one direction and r_2 to opposite side over M periods.

$$\Pi_s(\Theta) = \left(\Theta^{\frac{1}{2}} T_c^{-r_1} \Theta^{\frac{1}{2}} T_c^{-r_1} T_c^{r_2}\right)^M \tag{2.25}$$

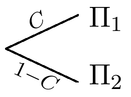
Swaps are usually agreed on without any initial premium. The expected profit from a swap is zero for both parties.

$$\Pi_s(\Theta)\, V_c = 0 \tag{2.26}$$

Swap agreements are known with a variety of payment modalities, concerning payment dates and interest rate specification. [Zag02]

2.4.2 Option

An option is defined by the alternatives among which can be selected and by the entity that does select. Feasible choices are specified by the two portfolios Π_1 and Π_2. Depending on the choice, one of the optional portfolios determines the remaining portfolio after the option expired. The deciding entity is characterized by her choice condition C. If C is one the first option Π_1 is chosen. If C is zero, Π_2 is the selected portfolio.



$$:= C\Pi_1 + (1 - C)\Pi_2 \tag{2.27}$$

The choice condition C is an operator. When applied to a value function V it returns an indicator function on the states that lead to choice Π_1. In the most common cases the value function itself carries the information on which choice is preferred. If the option is long, i.e. the holder of the product herself can choose, we can use the choice condition $C_{\max}$ to yield the scenario with higher value.

$$C_{\max} V = \Pi_1 V > \Pi_2 V = 1_{\Pi_1 V > \Pi_2 V} \tag{2.28}$$

In case of a short option, someone else can choose. Normally this leaves us with less valuable scenario $C_{\min}$.

$$C_{\min} V = \Pi_1 V < \Pi_2 V = 1_{\Pi_1 V < \Pi_2 V} \tag{2.29}$$

Bond option Combining the option and the transaction operator we can specify our first interest rate derivative. The operator term below describes an option on a zero coupon bond. Again, the term reads chronologically from left to right. First, we wait a time m and then choose between two scenarios. In the first scenario we first pay a strike price K and wait for the underlying maturity M until we receive a final redemption of 100. The second scenario is the identity operator and refers to no transaction.

$$\Theta^m \begin{array}{l} \overset{C}{\nearrow}\ T_c^{-K}\Theta^M T_c^{100} \\ \underset{1-C}{\searrow}\ . \end{array} \tag{2.30}$$

Asian option The Asian option, as denoted below, is easily read from left to right. First we initilialize an accounting variable a to have an initial value of 0. Then we repeat n times a time step that waits one period and then adds the current stock price S to a. Finally we can choose to receive the difference of the average stock price a/n and the strike price K as a cash payment.

$$\underset{a=0}{\text{大}} \left(\Theta T_a^S\right)^n \begin{array}{l} \overset{C}{\nearrow}\ T_c^{a/n-K} \\ \underset{1-C}{\searrow}\ . \end{array} \tag{2.31}$$

Random events

A special case of the option operator occurs if the choice condition is unobservable. Economy knows a lot of sudden and unforeseeable events. Companies can default and outstanding payments become void or unexpected damages have to be compensated. In this case we can use the same operation as for the option. The only difference is that we use a choice probability λ instead of the deterministic condition.

$$\begin{array}{l} \overset{\lambda}{\nearrow}\ \Pi_1 \\ \underset{1-\lambda}{\searrow}\ \Pi_2 \end{array} := \lambda\Pi_1 + (1-\lambda)\Pi_2 \tag{2.32}$$

Subdivision We assume an event that occurs with probability λ per period. If we want to decide whether the event occurred after a non unit time step of Δt we use the probability $\lambda^{\Delta t}$ instead of λ. For $\Delta t = 1/2$ we can easily verify that the event operator with $\sqrt{\lambda}$ is the square root of the event operator with probability λ.

$$\left(\begin{array}{l} \overset{\sqrt{\lambda}}{\nearrow}\ \cdot \\ \underset{1-\sqrt{\lambda}}{\searrow}\ X \end{array}\right)^2 V = \begin{array}{l} \overset{\sqrt{\lambda}}{\nearrow} \begin{array}{l} \overset{\sqrt{\lambda}}{\nearrow}\ V \\ \underset{1-\sqrt{\lambda}}{\searrow}\ X \end{array} \\ \underset{1-\sqrt{\lambda}}{\searrow}\ X \end{array} = \lambda V + (1-\lambda)X \tag{2.33}$$

Time continuous option

The time continuous option, also referred to as American option, is an option with the continuous right of exercise. Over a certain time span M the holder of the option has the continuous right to leave the product path and branch into the optional product Π_X. The option requires a time process that can be infinitely subdivided and an option that is exercisable after each infinitely small time step. The American option is defined by the limit of the number of time steps to infinity and the length of each step Δt to zero.

$$\lim_{\Delta t \to 0} \left(\Theta^{\Delta t} \begin{array}{l} \overset{C}{\nearrow}\ \Pi_X \\ \underset{1-C}{\searrow}\ . \end{array} \right)^{M/\Delta t} \tag{2.34}$$

As an alternative to the limit function we can write the same term with the differential form dt replacing the discrete time step Δt.

$$\left(\Theta^{dt} \begin{array}{l} \overset{C}{\nearrow}\ \Pi_X \\ \underset{1-C}{\searrow}\ . \end{array} \right)^{M/dt} \tag{2.35}$$

American stock option The American stock option consists of a cash transaction for the choice Π_X, which when executed ends the investment. First, the amount $S - K$ is transfered. Then the value function V ends the operator term. The function can be considered as a constant operator that evaluates V regardless of what function it is applied to.

$$\Pi_X = T_c^{S-K} V \tag{2.36}$$

Space continuous option

Space continuous options require continuous values to fully describe the selected action. A portfolio manager has the option to sell or buy more or less arbitrary amounts of stocks in each period. The objective of the action might be a rebalancing of the portfolio or an adjustment of a hedge position. Assume an action operator A for an activity, that can be repeated arbitrarily often in instantaneous time. Let $A^\star$ be the optimal exercise of A with respect to utility U:

$$A^\star = A^{x^*} \tag{2.37}$$

Whereas x^* is the optimal operator power with respect to utility operator U.

$$x^* = \operatorname{argmax} U A^{x^*} \tag{2.38}$$

Hedging A typical application of the space continuous option is the optimal rehedge, where we can buy or sell an arbitrary number of stocks. The operator that buys one stock increases our deposit h by one and decreases our cash account by the current stock price S.

$$A = T_c^{-S} T_h \tag{2.39}$$

For an optimal hedge investment we have to apply the stock buying operator A with the optimal exponent.

$$A^\star = \left(T_c^{-S} T_h\right)^\star \tag{2.40}$$

For a function V that depends on c, h and some variables x the operator that buys n stocks can be solved explicitly.

$$A^n\, V(c,h,\mathrm{x}) = V(c - nS, h + n, \mathrm{x}) \tag{2.41}$$

The result of the optimal investment $A^\star$ is then obtained by maximizing the utility UA^nV over the number of bought stocks n.

2.5 Pricing via arbitrage

This section gives a quick introduction into arbitrage pricing to find the unique option price in a single step binomial tree. We will try to replicate the pay off of a call option precisely. In fact, there exists an investment strategy that yields the same final cash value as a call option.
In order to set up our strategy we write the usual operator sequence in chronological order from left to right. First we buy x stocks at price S, thus withdrawing xS from our account c. Then we buy one option at a price y. Then we wait one period, sell all our stocks and exercise the option.

$$\Pi(\Theta) = \underbrace{T_c^{-\mathrm{x}S}\, T_c^{-\mathrm{y}}}_{\substack{\text{buy x stocks}\\ \text{at } S \text{ and } 1\\ \text{option at y.}}} \underbrace{\Theta}_{\text{wait}} \underbrace{T_c^{\mathrm{x}S}}_{\substack{\text{sell}\\ \text{stocks}}} \underbrace{\begin{array}{l} \xrightarrow{\max} T_c^{S-K} \\ \longrightarrow \; . \end{array}}_{\text{exercise option}} \tag{2.42}$$

2.5.1 The process

Now we need to define what happens during the time of our inactivity. The time process consists of two effects. First, an interest rate of size rc is payed to the cash account c. And, second, two different branches are taken with probabilities p and $1-p$. In the first case S is increased by $+S$, resulting in $2S$. The other possible outcome reduces S to its half.

$$\Theta = T_c^{rc} \begin{array}{l} \xrightarrow{p} T_S^{+S} \\ \xrightarrow[1-p]{} T_S^{-\frac{1}{2}S} \end{array} \tag{2.43}$$

Initial values The initial values of the process parameters are always inserted after the complete operator term is expanded. We will use the operator 大 to indicate this insertion. 大 is always the leftmost operator in a sequence, since the variables do no longer occur in the function after its application.

$$大 V = V\Big|_{\substack{S=100 \quad c=0\\ K=150 \quad r=1/9}} \tag{2.44}$$

The operator 大 inserts the initial values and triggers a computational evaluation procedure, if necessary. Binomial tree models with short time horizons can normally be handled symbolically by computer algebra systems and thus allow the automated extraction of many implicit parameters.

2.5.2 Arbitrage-free price

In order to find the fair option price y, we have to find a hedge position x such that our final cash amount is zero under all conditions. Basically, we write our evaluation formula chronologically. First we fix the initial values, then go through our investment strategy Π and finally query our cash value V_c, with $V_c = c$ from (2.4). This operator sequence evaluates the expected amount π of account c after the completion of this strategy.

$$\begin{aligned}\pi &= \text{大}\,\Pi(\Theta)V_c \qquad (2.45)\\ &= \text{大}\,T_c^{-\mathrm{x}S}\,T_c^{-\mathrm{y}}\Theta T_c^{\mathrm{x}S}(c+\max(S-K,0))\\ &= \text{大}\,T_c^{-\mathrm{x}S}\,T_c^{-\mathrm{y}}\Theta\,(c+\mathrm{x}S+\max(S-K,0))\\ &= \text{大}\,T_c^{-\mathrm{x}S}\,T_c^{-\mathrm{y}}\big(c(1+r)+p\,(\mathrm{x}2S+\max(2S-K,0))+\\ &\quad (1-p)\,(\mathrm{x}S/2+\max(S/2-K,0))\,\big)\\ &= \text{大}\big((c-\mathrm{x}S-\mathrm{y})(1+r)+p\,(\mathrm{x}2S+\max(2S-K,0))+\\ &\quad (1-p)\,(\mathrm{x}S/2+\max(S/2-K,0))\,\big)\end{aligned}$$

Inserting the initial values according to the configuration of 大 (2.44) this yields:

$$p\left(50+\frac{800}{9}\mathrm{x}-\frac{10}{9}\mathrm{y}\right)+(1-p)\left(-\frac{550}{9}\mathrm{x}-\frac{10}{9}\mathrm{y}\right) \qquad (2.46)$$

The final cash value is the same as has been derived in [Ros99]. The difference is that this computation follows straight forward mathematical expansions. You should keep in mind that more realistic examples with multiple steps and additional assets quickly lead to algebraic results that can span several pages. A simple and compact calculus to command a computer algebra system is therefore essential.

Fortunately, this result is short and several methods can be used to find the solution. The equation system that is to solve requires that the profit π is zero for all probabilities p.

$$\exists \mathrm{x},\mathrm{y}:\ \forall p:\pi=0 \qquad (2.47)$$

We can turn this into a finite system of equations, by inserting different values for p and verify the result. In the above example we retrieve an option price of 55/3 and a hedge position of −1/3 stocks.

$$\widetilde{\mathrm{x}}=-\frac{1}{3},\quad \widetilde{\mathrm{y}}=\frac{55}{3} \qquad (2.48)$$

The existence of a solution is not always guaranteed. The fact that we do have a valid single solution is due to the fact that we deal with a so called complete market, in which all options can be replicated by a unique stock trading strategy [Zag02, p62].

2.5.3 Equivalent martingale measure

A simplified and efficient method for finding the same price is done with the equivalent martingale measure. There exists a pseudo probability $\widetilde{p}$ for which the discounted stock price is a martingale, i.e. the discounted expected value tomorrow is equal to todays' value.

$$\exists p : \frac{\Theta S}{1+r} = S \tag{2.49}$$

Which expands to an equation for p, after inserting the definiton of Θ (2.43).

$$\frac{2pS + \frac{1}{2}(1-p)S}{1+\frac{1}{9}} = S \tag{2.50}$$

The result is found easily.

$$\widetilde{p} = \frac{11}{27} \tag{2.51}$$

With the new value for p we can create a transformed process $\widetilde{\Theta}$ that evaluates the expected value under the equivalent martingale measure, where stock prices are expected to grow with the interest rate.

$$\widetilde{\Theta} = \Theta|_{p=\widetilde{p}} \tag{2.52}$$

This transformed operator can now be directly applied to the pay off structure of the option to compute the fair price, exactly as we did it in the Black&Scholes model (see 2.3.3).

$$\text{大}\, \frac{\widetilde{\Theta} \max(S-K,0)}{1+r} = \frac{55}{3} \tag{2.53}$$

The evaluation operator 大 causes the computational system to switch to a numerical scheme after the full operator term was specified.

2.6 Example

In this final example we consider a portfolio manager who periodically rebalances her hedge portfolio. Our trader has an obligation to her customers in the form of a call option with strike 10. Suppose her mission was to optimally meet her obligation in either cash or stocks with a minimum squared distance. We do not work in complete markets, since we assume trading opportunities at discrete times and will later introduce market impact. Thus all hedges will bear at least some risk. Our function V contains the final wealth and its square.

$$V = \begin{pmatrix} V_a - V_{call} \\ -(V_a - V_{call})^2 \end{pmatrix} = \begin{pmatrix} c + Sh - \max(S-10,0) \\ -\big(c + Sh - \max(S-10,0)\big)^2 \end{pmatrix} \tag{2.54}$$

For a least square hedge we define the utility operator U as the second component of vector V.

$$UV = V_2 \tag{2.55}$$

We consider a single asset market where the price level S fluctuates according to a Brownian motion. The process operator is for now a single $\mathcal{B}$ in the direction of S, thus modelling a variation of S with volatility $\sigma = 1$. (2.15).

$$\Theta = B_S^1 \tag{2.56}$$

The numeric solution to the expected final amount of cash and the expected utility is done by the numerical evaluation scheme 大. This operator evaluates an approximation to the operator term ΘV and inserts the initial values for S, c and h. Depending on the chosen method 大 initializes a scenario in a Monte-Carlo simulation or retrieves the initial position in a PDE result.

$$\text{大}\Theta V = \Theta V\Big|_{\substack{S=10\\c=0.4\\h=0}} = \begin{pmatrix} 0 \\ -0.34 \end{pmatrix} \tag{2.57}$$

The first component tells us that the trader meets the expectation of her obligation precisely. Hence, the initial cash value of 0.4 is the expected value for the option. The low utility in the second component reveals the high risk inherent in holding the unhedged option.

2.6.1 Hedging activity

Now we want to see, if the utility can be increased by trading in the underlying stock with a reoptimization frequency of Δt. The operator that buys one stock subtracts the current stock price from cash account c and adds one stock the deposit h. The $\star$ indicates the optimal exponent.

$$\Pi(\Theta) = \left(\left(T_c^{-S}\, T_h\right)^{\star}\ \Theta^{\Delta t}\right)^{\frac{1}{\Delta t}} \tag{2.58}$$

According to our numeric results, the utility increases significantly with a portfolio rebalancing frequency of $\Delta t = 1/4$. The expected profit is still zero. The strategy produces no extra costs.

$$\text{大}\Pi(\Theta)V = \begin{pmatrix} 0 \\ -0.03 \end{pmatrix} \tag{2.59}$$

Complete market With our choice for the process Θ the risk of every obligation V can be reduced to zero by infinitely many rehedges. Markets governed by such processes are called complete markets [Shr97].

$$\lim_{\Delta t \to 0} \text{大}\Pi(\Theta)V = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{2.60}$$

2.6.2 Supply and demand

Due to the law of supply and demand real stock prices vary with the traded amount. Individual market participants will enter in the order book the prices at which they are willing to buy or sell. The more stocks we want to trade, the more people we have to satisfy and the worse is our price. We assume a linear order book in which every transaction has a price impact of κ per stock. The new strategy Π_I performes $1/\Delta t$ rehedges and considers the price impact on S.

$$\Pi_I(\Theta) = \left(\left(T_S^{\kappa}\, T_c^{-S}\, T_h\right)^{\star}\ \Theta^{\Delta t}\right)^{\frac{1}{\Delta t}} \tag{2.61}$$

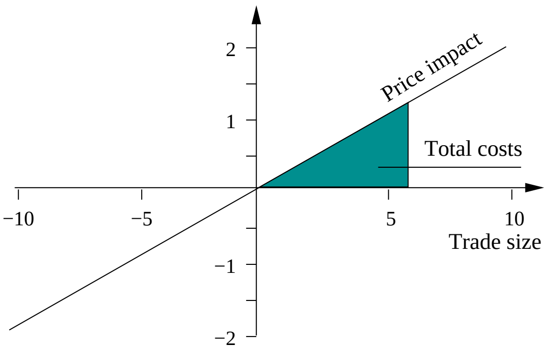


Figure 15: In the linear market impact model every traded stock shifts the price by $\kappa = 0.2$. The total cost, marked by the filled area, is proportional to the square of the trade size.

Applied to our valuation function this reveals the expected final cash amount and the utility. The assumed market impact ratio, sometimes defined as market elasticity, is $\kappa = 0.2$. Numeric evaluation can be performed with PDE methods or with our operators from chapter 1.

$$\text{大}\Pi_I(\Theta)V \begin{pmatrix} -0.05 \\ -0.04 \end{pmatrix} \tag{2.62}$$

Our trader is expected to lose 0.05 units of cash due to market friction. These are not transaction costs, since the selling operator is the exact inverse of the buying operator. The costs originate from procyclic trading and are gained by the anticyclic investor. If our trader wanted to reduce her loss then she had to take more risk. With respect to her quadratic utility function the presented values are optimal.

2.6.3 Market impact

Finally we might ask for the market impact of the hedging strategy on the stock price. The valuation function V is easily extended by additional components for the expected stock price S and its square.

$$\text{大}\Pi_I(\Theta) \begin{pmatrix} V \\ S \\ S^2 \end{pmatrix} = \begin{pmatrix} \Pi_I(\Theta)V \\ 10.117 \\ 103.5 \end{pmatrix} \tag{2.63}$$

The first interesting result is that the stock price S is expected to rise by 1.17%, which is due to the fact that we bought stocks for hedging purposes but did not necessarily sell them finally. Economists will refer to this phenomenon as inflation, induced by a 40 cent increase of circulating cash and a corresponding overdemand on the stock exchange. The second parameter needs some treatment to reveal its information.

$$\tilde{\sigma} = \sqrt{\mathbb{E}(S^2) - \mathbb{E}(S)^2} = \sqrt{103.5 - 10.117^2} = 1.08 \tag{2.64}$$

We remember that the volatility σ was initialized to one in (2.56) and now increased to 1.08. The final result is that our hedging strategy is procyclic and increases the market volatility of the stock by 8%. This is a realistic value for very large investments.

2.7 Conclusion

This chapter introduced an operator term Π for the sequence of human investment activities. Three kinds of operations were considered to constitute the space of possible strategies. The first kind of occupation is inactivity. Whenever Θ occurs in an operator term it refers to a period of passive observation. During that time, external state variables may vary according to a stochastic process [Shr97, KO01] or as described by a partial differential equation [TR00]. The second possible activity is a transaction. The operator T initiates a deterministic effect on our parameter set. Typical instances are the transfer of goods or cash. The third and final operation models an option. Multiple operator terms $\Pi_1 \cdots \Pi_n$ can be offered as choices for further procedure. The decision criteria can be based on the current state and the expected values for each choice.

The operator term Π is written in chronological order from left to right and makes use of some mathematical concepts like the operator power for repeated actions. Solutions to risk measures and expected values can be evaluated directly with either a numerical or in some instances symbolic method. Thereby the operators provide a much more natural procedure to derive the formulas that solve a problem, without leading to any new solutions. The computational evaluation of operator terms requires algorithms that work on the domain of mathematical functions. A framework for an appropriate data structure was defined in the previous chapter. The subsequent chapter 3 shows how common definitions for Θ can be expressed by sequences of operators that can be approximated by the B and T operators.

3 The Theta-notation for stochastic processes

This chapter presents a notation for stochastic processes that is based on the foundation of operator theory. Any stochastic Markov process can be written as an operator Θ evaluating an expected value of an arbitrary operand function under a given process. Based on this representation, any stochastic process and any statistic measure can be described in operator form. We will see various applications in the field of quantitative finance.

The operator notation has two main advantages over the prevailing stochastic analysis. Firstly, the process itself as well as statistical measures are written in explicit form and can be evaluated by plain algebraic expansions or explicitly determined numerical methods. This considerably simplifies the process of implementing a stochastic model in a computer algebra system or any numerical package. Secondly, the notation allows for chronological notations of business events and trading strategies. This is useful, since many financial products are currently not representable in closed form or as an explicit mathematical expression.

3.0 Introduction

Stochastic processes are used in quantitative finance, economy and various fields of physics for the description of random time series. A mathematically rigid theory was developed and applied in quantum mechanics to describe the evolution of a particle that moves randomly or, at least, in a way that is not deducible from observable information [Hua98]. Random walks are of crucial importance in finance, since all market parameters and prices are assumed to be unpredictable. Financial theories postulate the stochastics of future states by the means of a stochastic process.

Two competing mathematical methods for the definition and the analysis of stochastic processes were introduced by physicists. One was based on differential operators and used functions to describe a current state. The other was based on stochastics and measure theory, where the current state was defined by a vector of random variables. The fundamental relationship between both approaches was developed by Feynman-Kac [Zag02, p40]. Since then it was possible to convert both formulations into each other. Despite high compatibility there have not been any serious attempts to develop a larger framework for financial processes based on differential equations and functional operators.

This document suggests a complementary Θ-notation for stochastic processes in the context of financial applications. Despite the apparent domination of stochastic calculus there seem to be important advantages of the operator-based notation. The first and possibly the key argument in favor of operator calculus is the existence of explicit expressions for contract details and trading strategies including embedded options, minimum guarantees, event triggers and non-delta hedges [Dir04]. This is in fact a decisive feature since financial portfolio details are currently only written in text and prose form. Mathematically explicit expressions were currently confined to very specific portfolio properties and to a fixed stochastic process. The second advantage concerns practical applicability. Given a process in operator form there are rather explicit methods to extract its statistical properties as they can be performed by a computer algebra system. Finally, the Θ-notation deals much more naturally with the many financial processes that do not live in continuous time, such as free form probability distributions or some regime switching models. The Θ-calculus should be considered as highly complementary to stochastic calculus. The theoretical framework around stochastic processes is much more established and many classical models do have more compact and nicer formulas in stochastic calculus.

3.1 Stochastic model

In this section we derive the stochastic framework for a process X and its process operator Θ. We plan to define an operator Θ that fully describes the algebraic properties of the stochastic process X, but can do without any reference to stochastic calculus. On our way to turn any stochastic process into operators we start with an investigation of general Markov processes. Consider a probability space $(\Omega, \mathfrak{A}, P)$ that determines the probability distribution of the random seed ω. At every time t the process X satisfies a stochastic differential equation.

$$\begin{aligned} X &: \left(\mathbb{R}_0^+ \times \Omega \to \mathbb{R}^n\right) \\ d_t X(t, \omega) &= F(X(t), \omega, dt) \end{aligned} \tag{3.1}$$

The instantaneous change of the process variable X is determined by a function F with three parameters. The first parameter is the current process state, that is sufficient to determine all deterministic process components. The stochastic variable ω allows the process dependence on random events. The final parameter dt is the time resolution of the process evaluation, that is required for fractal processes such as the Brownian motion.

To pick up our operators for financial contracts we will briefly repeat the use of V (see 2.2) and Θ (see 2.8) in our new framework. We consider an evaluating function V that gives us some interpretation of the current process state. This can be an event indicator or some utility that we can draw from the process.

$$V : \mathbb{R}^n \to \mathbb{R}^m \tag{3.2}$$

With the operator Θ we can look one step Δt into the future and evaluate the expectation of our function V under the future process state. Since we exclusively consider Markov processes, and those that can be transformed into Markovian form, the definition for Θ does not depend on time t.

$$\begin{array}{rcl} \Theta \ : \ (\mathbb{R}^n {\to} \mathbb{R}^m) & \to & (\mathbb{R}^n {\to} \mathbb{R}^m) \\ \Theta^{\Delta t} V(\mathrm{x}) & := & \mathbb{E}\left(V\big(X(t+\Delta t)\big) \middle| X(t) = \mathrm{x} \right) \end{array} \tag{3.3}$$

Based on the operator $\Theta^{\Delta t}$ and its application to arbitrary but well selected functions we can solve a wide range of expected value problems, in particular those associated with present values and risk measures in finance.

3.1.1 Transition probabilities

A common tool for the definition of stochastic processes is the probability density. After every time step the current state x changes randomly. Let $p_{\Delta t}(\mathrm{x}, \mathrm{x}')$ be the probability density for the state to travel from x to x′ within a time step of Δt. The operator $\Theta^{\Delta t}$ is defined as a convolution with $p_{\Delta t}$.

$$\Theta^{\Delta t} V(\mathrm{x}) = \int_{\mathbb{R}^n} p_{\Delta t}(\mathrm{x}, \mathrm{x}') V(\mathrm{x}') d\mathrm{x}' \tag{3.4}$$

Every process has a probability density, when we take extended functions into account. The delta function is well defined as a Dirac sequence and integrates well into calculus.

Lemma: We can apply the Θ operator to a delta function as bellow and extract the probability density.

$$p_{\Delta t}(\mathrm{x}, \mathrm{x}') = \Theta^{\Delta t} \delta(\mathrm{x} - \mathrm{x}') \tag{3.5}$$

Proof: We have to prove that the extracted probability density (3.5) correctly reproduces the Θ operator when inserted into the convolution formula (3.4). The proof utilizes the linearity of Θ, which is due to the linearity of the expected value (3.3), and exploits Θ's independence of the temporary variable x′.

$$\begin{array}{rcl} \displaystyle\int_{\mathbb{R}^n} p_{\Delta t}(\mathrm{x}, \mathrm{x}') V(\mathrm{x}') d\mathrm{x}' & \overset{(3.5)}{=} & \displaystyle\int_{\mathbb{R}^n} \left(\Theta^{\Delta t} \delta(\mathrm{x} - \mathrm{x}')\right) V(\mathrm{x}') d\mathrm{x}' \\ & = & \Theta^{\Delta t} \displaystyle\int_{\mathbb{R}^n} V(\mathrm{x}') \delta(\mathrm{x} - \mathrm{x}') d\mathrm{x}' \\ & = & \Theta^{\Delta t} V(\mathrm{x}). \end{array} \tag{3.6}$$

3.1.2 Deterministic transition

A simple, but common, transition is a deterministic step from a current state X to a deterministic new state $f(X)$. The corresponding density is the delta function $p(\mathrm{x}, \mathrm{x}') = \delta(f(\mathrm{x}) - \mathrm{x}')$. We use the special symbol 大 to indicate such a step.

$$\underset{\mathrm{x}=f(\mathrm{x})}{大} V \ := \ V|_{\mathrm{x}=f(\mathrm{x})} \tag{3.7}$$

There is a notational difference of the 大 operator and the more common expression using the bar operator, as indicated by the right hand side of the definition 3.7. The 大 operator is written left of the argument V, which allows us to write different effects, as introduced in chapter 2, chronologically from left to right. The standard application of this operator is to provide initial values to the process parameters. This operation is always the leftmost operator in a sequence. After its application all process variables are replaced by numbers. (see 2.1)

$$\text{大} := \underset{\mathrm{x}=X(0)}{\text{大}} \tag{3.8}$$

3.2 Differential processes

Differential equations are a well established technique to describe the evolution of physical settings over time. Physics and finance have many similarities that already triggered the scientific research field of econophysics. The most vivid example is the particle in heated media that moves according to the same laws as a stock price in volatile markets. We will focus on two special cases of a partial differential equation and define appropriate operators that solve these equations. The first PDE is the transport equation that models a deterministic evolution of the process parameters. The second is the heat equation that models random state changes according to a Brownian motion.

Definition In its most general form a differential process operator O solves a partial differential equation that is defined by a differential operator D. Let this D depend on the value of the function f and its spatial derivatives up to order two.

$$\begin{aligned} D \;:\; (\mathbb{R}^n \to \mathbb{R}^m) &\;\to\; (\mathbb{R}^n \to \mathbb{R}^m) \\ Df(\mathrm{x}) &\;=\; \mu(\mathrm{x})\nabla f(\mathrm{x}) + \frac{1}{2}\nabla^{\mathsf{T}}\Sigma(\mathrm{x})\nabla f(\mathrm{x}) \end{aligned} \tag{3.9}$$

Which can be expanded into a sum of derivatives:

$$Df(\mathrm{x}) = \sum_{i=1}^{n}\mu_i(\mathrm{x})\frac{\partial}{\partial \mathrm{x}_i}f(\mathrm{x}) + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\Sigma_{ij}(\mathrm{x})\frac{\partial^2}{\partial \mathrm{x}_i \partial \mathrm{x}_j}f(\mathrm{x}) \tag{3.10}$$

The operator O can be defined by the solution of a partial differential equation generated by D. We choose the time variable t to play the role of time in classical PDE analysis.

$$\frac{\partial}{\partial t}O^t = DO^t \tag{3.11}$$

The definition above can also be written in a long version, where functions and their arguments are appended explicitly. Please note that equation (3.11) is just a short form of (3.12).

$$\begin{aligned} \forall V : \exists \tilde{V} : \qquad \tilde{V}(0,\mathrm{x}) &= V(\mathrm{x}) \\ \tilde{V}(t,\mathrm{x}) &= O^t\, V(\mathrm{x}) \\ \frac{\partial}{\partial t}\tilde{V}(t,\mathrm{x}) &= D\,\tilde{V}(t,\mathrm{x}) \end{aligned} \tag{3.12}$$

With this PDE we can solve the spacial part of the Coshy problem [Zag02, p38] and will later include the means to express the discounting drift.

Lemma: The superposition that is written to the operator O satisfies the power rules and thus is an exponent of the operator. There are three properties of the operator power, which will be used later in this document.

$$\begin{align} O^a O^b &= O^{a+b} \tag{3.13} \\ O^0 &= Id \tag{3.14} \\ (O^a)^b &= O^{ab} \tag{3.15} \end{align}$$

Proof: First, we verify the power rule (3.13). Assume $\widetilde{V}_1$ and $\widetilde{V}_2$ solve the PDE (3.12).

$$\begin{align} \widetilde{V}_1(0,\mathrm{x}) = V(\mathrm{x}) &\Rightarrow O^b = \widetilde{V}(b,\mathrm{x}) \tag{3.16} \\ \widetilde{V}_2(t,\mathrm{x}) = \widetilde{V}_1(t+b,\mathrm{x}) &\Rightarrow \widetilde{V}_2(0,\mathrm{x}) = O^b V(\mathrm{x}) \notag \\ &\Rightarrow \widetilde{V}_2(a,\mathrm{x}) = O^a O^b V(\mathrm{x}) = \widetilde{V}_1(a+b,\mathrm{x}) = O^{a+b} V(\mathrm{x}) \notag \end{align}$$

Proving (3.14) is almost trivial. Again, we assume $\widetilde{V}$ to solve the PDE (3.12).

$$\widetilde{V}(0,\mathrm{x}) = V(\mathrm{x}) \Rightarrow O^0 V(\mathrm{x}) = V(\mathrm{x}) \tag{3.17}$$

The final power rule (3.15) is a direct consequence of the the first two rules. For $b \in \mathbb{N}$ it is just a rewrite of (3.13) with multiple summands. For $1/b \in \mathbb{N}$ the operator root O^b is correct due to $(O^b)^{1/b} = Id$. Hence, all rational number are proved. Irrational numbers follow from the continuous property of the PDE solution.

3.2.1 Drift operator

The drift operator $\mathcal{T}^{\mu}$ solves the so-called transport equation over the unit time interval with a velocity field μ, according to (3.9) with $\Sigma = 0$. Possible financial meanings are the transfer of goods and assets or a deterministic drift in observable parameters. Technical applications of $\mathcal{T}^{\mu}$ include the introduction of temporary variables and domain transformations.

$$\begin{align} \mathcal{T}^{\mu} : (\mathbb{R}^n \to \mathbb{R}^m) &\to (\mathbb{R}^n \to \mathbb{R}^m) \tag{3.18} \\ \frac{\partial}{\partial t}\mathcal{T}^{t\mu} &= \mu(\mathrm{x}) \nabla \mathcal{T}^{t\mu} \notag \end{align}$$

Univariate projection In most situations, the drift operator $\mathcal{T}^{\mu}$ applies in only one direction, thus solving only a univariate partial differential equation. The considered dimension is written as an index.

$$\begin{align} \mathcal{T}_{\mathrm{x}}^{\mu} : (\mathbb{R}^n \to \mathbb{R}^m) &\to \mathbb{R}^n \to \mathbb{R}^m \tag{3.19} \\ \frac{\partial}{\partial t}\mathcal{T}_{\mathrm{x}}^{t\mu} &= \mu(\mathrm{x}) \frac{\partial}{\partial \mathrm{x}} \mathcal{T}_{\mathrm{x}}^{t\mu} \notag \end{align}$$

Properties

The resulting drift operator can usually be solved analytically. The correctness of the solutions below are easily verified by differentiation with respect to the index variable x according to (3.12).

Lemma: In the following there are the solutions for constant and linear exponents. For real valued exponents, the $\mathcal{T}$ operator is equivalent to T, defined in section 2.4.1.

$$\begin{aligned} \mathcal{T}^{a} V(\mathrm{x}) &= V(\mathrm{x}+a) \qquad \forall a \in \mathbb{R} && (3.20)\\ \mathcal{T}^{a\mathrm{x}} V(\mathrm{x}) &= V(e^{a}\mathrm{x}) \qquad \forall a \in \mathbb{R} && (3.21) \end{aligned}$$

These two results account for probably most instances of the $\mathcal{T}$ operator.

Proof: The proof is briefly sketched below. We use the function $V(\mathrm{x}+t)$ and $V(\mathrm{x}e^{t})$ as $\widetilde{V}$ and verify the equations from (3.12). The results in the lemma represent the special case of $t = 1$ for other values of t the operator can be raised to the power of t (see (3.15)).
$\frac{\partial}{\partial t} V(\mathrm{x}+at) = aV'(\mathrm{x}+at) = a\frac{\partial}{\partial \mathrm{x}} V(\mathrm{x}+at)$ and
$\frac{\partial}{\partial t} V(\mathrm{x}e^{at}) = a\mathrm{x}e^{at} V'(\mathrm{x}e^{at}) = a\mathrm{x}\frac{\partial}{\partial \mathrm{x}} V(\mathrm{x}e^{at})$

Lemma: The drift operator commutes with function evaluation $\mathcal{T}^{\mu} f(\mathrm{x}) = f(\mathcal{T}^{\mu}\mathrm{x})$. Hence, it can also be written as a deterministic step (see 3.1.2).

$$\begin{aligned} \mathcal{T}^{\mu} V(\mathrm{x}) &= \underset{\mathrm{x}=\mathcal{T}^{\mu}\mathrm{x}}{\text{太}} V(\mathrm{x}) && (3.22)\\ &= V(\mathcal{T}^{\mu}\mathrm{x}) \end{aligned}$$

Proof: The proof is briefly sketched below and is based on the complete differential and the chain rule.
$\frac{\partial}{\partial t} V(\mathcal{T}^{t\mu}\mathrm{x}) = (\frac{\partial}{\partial t}\mathcal{T}^{t\mu})\nabla V(\mathrm{y})|_{y=\mathcal{T}^{t\mu}\mathrm{x}} = (\mu(\mathrm{x})\nabla\mathcal{T}^{t\mu}\mathrm{x})\nabla V(\mathrm{y})|_{y=\mathcal{T}^{t\mu}\mathrm{x}} =$
$= \mu(\mathrm{x})\nabla V(\mathcal{T}^{t\mu}\mathrm{x})$

Lemma: With the previous lemma the $\mathcal{T}$ operator can be converted into an ordinary differential equation, for which an integral representation is known. The exponent μ must be continuous and non zero to yield a unique solution.

$$\begin{aligned} \mathcal{T}^{\mu(\mathrm{x})}\mathrm{x} &= f_{\mathrm{x}}(1) && (3.23)\\ &\Leftarrow \; f_{\mathrm{x}}^{-1}(\mathrm{y}) = \int_{\mathrm{x}}^{\mathrm{y}} \frac{1}{\mu(h)} dh \end{aligned}$$

Proof: Again, we proof the more general case $\mathrm{y} = \mathcal{T}^{t\mu}\mathrm{x} = f_{\mathrm{x}}(t)$. From the explicit formula for $t = f_{\mathrm{x}}^{-1}(\mathrm{y})$ can conclude the derivative of $f(t)$, according to the differentiation rule for inverse functions
$\frac{\partial}{\partial t}\mathcal{T}^{t\mu}\mathrm{x} = \frac{\partial}{\partial t} f_{\mathrm{x}}(t) = \frac{1}{\frac{\partial}{\partial \mathrm{y}} f_{\mathrm{x}}^{-1}(\mathrm{y})} = \mu(\mathrm{y}) = \frac{\mu(\mathrm{y})\mu(\mathrm{x})}{\mu(\mathrm{x})} = \mu(\mathrm{x})\frac{\partial}{\partial \mathrm{x}} f(t)$

Example

This example models the growth of a money market account with time dependent interest rates. We are interested in the expected value of a function $V(c,t)$ that depends on the cash account c and time t. The process Θ consists of two operators. One updates the time variable t. The other pays the interest rate r_t.

$$\begin{aligned} \mathcal{T}_t V(c,t) &\overset{(3.20)}{=} V(c,t+1) \\ \mathcal{T}_c^{cr_t} V(c,t) &\overset{(3.21)}{=} V(ce^{r_t},t) \end{aligned} \tag{3.24}$$

The future value can be computed by repeated application of our process operators to the cash and time sensitive function V.

$$\begin{aligned} (\mathcal{T}_c^{cr_t}\mathcal{T}_t)^M V(c,t) &= (\mathcal{T}_c^{cr_t}\mathcal{T}_t)^{M-1} V(ce^{r_t},t+1) \\ &= (\mathcal{T}_c^{cr_t}\mathcal{T}_t)^{M-2} V(ce^{r_t}e^{r_{t+1}},t+2) \\ \cdots &= V\left(c\prod_{p=0}^{M-1} e^{r_{t+p}},t+M\right) \end{aligned} \tag{3.25}$$

Inserting the initial cash amount for c in the solution yields the functions final value after M periods.

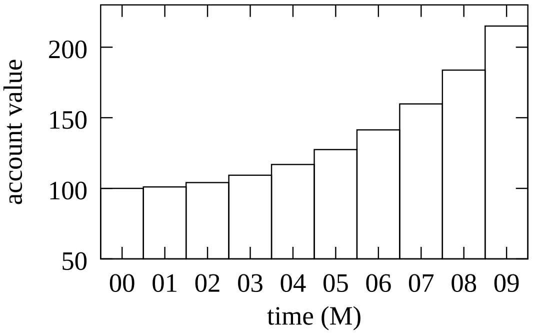


Figure 16: Growth of an interest rate account with time dependent rate r_t.

Plotted function:
$$f(M) = \underset{\substack{c=100\\t=0}}{\text{大}} \Theta^M V$$

$$\text{with} \quad \begin{aligned} \Theta &= \mathcal{T}_c^{cr_t}\mathcal{T}_t \\ r_t &= 1\% + 2\% t \\ V &= c \end{aligned}$$

3.2.2 Blur operator

The blur[1] operator $\mathcal{B}^{\Sigma}$ solves the heat equation, according to (3.9) with $\mu = 0$, which has a Brownian motion as source of randomness. Heated particles move according to a Brownian motion, which is unpredictable and random. The random movements in each direction are modeled to have the covariance matrix Σ. Financial models use the blur operator as the main source of uncertainty.

$$\begin{aligned} \mathcal{B}^{\Sigma} : (\mathbb{R}^n \to \mathbb{R}^m) &\to (\mathbb{R}^n \to \mathbb{R}^m) \\ \frac{\partial}{\partial t}\mathcal{B}^{t\Sigma} &= \frac{1}{2}\nabla^{\mathsf{T}}\Sigma(\mathrm{x})\nabla\mathcal{B}^{t\Sigma} \end{aligned} \tag{3.26}$$

---

[1] "Blur" is the name given to this operation in image processing, due to a function's smoother appearance after the operation.

Univariate projection In most situations, the blur operator $\mathscr{B}$ applies in only one direction, solving only the univariate heat equation. The uncertain dimension can be written as an index to the operator.

$$\begin{aligned} \mathscr{B}_{\mathrm{x}}^{\sigma^2} : (\mathbb{R}^n \to \mathbb{R}^m) &\to \mathbb{R}^n \to \mathbb{R}^m \\ \frac{\partial}{\partial t} \mathscr{B}_{\mathrm{x}}^{t\sigma^2} &= \frac{\sigma^2(\mathrm{x})}{2} \frac{\partial^2}{\partial \mathrm{x}^2} \mathscr{B}_{\mathrm{x}}^{t\sigma^2} \end{aligned} \tag{3.27}$$

The univariate σ is the standard deviation of the modeled variable x. The time normalized exponent $\frac{\sigma}{\sqrt{\Delta t}}$ was termed volatility in finance. The subsequent sections will derive solutions for specific classes of exponents.

Real exponent

The solution of the blur operator with real exponents is expressed by the convolution with the Gaussian density function. We insert the bell curve for $p_{\Delta t}$ in (3.4) from section.

$$\begin{aligned} \mathscr{B}_{\mathrm{x}}^{\sigma^2} V(\mathrm{x}) &= \frac{1}{\sigma\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left(-\frac{1}{2}\left(\frac{\mathrm{x}-\mathrm{y}}{\sigma}\right)^2\right) V(\mathrm{y}) d\mathrm{y} \\ &\overset{u=\frac{\mathrm{x}-\mathrm{y}}{\sigma}}{=} \frac{1}{\sigma\sqrt{2\pi t}} \int_{\mathbb{R}} e^{-u^2/2} V(\mathrm{x}-\sigma u) du \end{aligned} \tag{3.28}$$

Proof: Now we shall proof the equivalence of the PDE definition and the convolution formula. The proof covers the more general case with $t\sigma^2$ in the exponent.

$$\begin{aligned} \frac{\partial}{\partial t} \mathscr{B}^{t\sigma^2} V(\mathrm{x}) &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \frac{\partial}{\partial t} \frac{1}{\sigma\sqrt{t}} \exp\left(-\frac{1}{2}\frac{(\mathrm{x}-\mathrm{y})^2}{t\sigma^2}\right) V(\mathrm{y}) d\mathrm{y} \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \left(\frac{(\mathrm{x}-\mathrm{y})^2}{2\sigma^3 t^2\sqrt{t}} - \frac{1}{2\sigma t\sqrt{t}}\right) \exp\left(-\frac{1}{2}\frac{(\mathrm{x}-\mathrm{y})^2}{t\sigma^2}\right) V(\mathrm{y}) d\mathrm{y} \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \frac{1}{\sigma\sqrt{t}} \left(\frac{(\mathrm{x}-\mathrm{y})^2}{2t^2\sigma^2} - \frac{1}{2t}\right) \exp\left(-\frac{1}{2}\frac{(\mathrm{x}-\mathrm{y})^2}{t\sigma^2}\right) V(\mathrm{y}) d\mathrm{y} \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} -\frac{\partial}{\partial \mathrm{x}} \frac{1}{\sigma\sqrt{t}} \frac{(\mathrm{x}-\mathrm{y})}{2t} \exp\left(-\frac{1}{2}\frac{(\mathrm{x}-\mathrm{y})^2}{t\sigma^2}\right) V(\mathrm{y}) d\mathrm{y} \\ &= \frac{\sigma^2}{2} \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \frac{\partial^2}{\partial \mathrm{x}^2} \frac{1}{\sigma\sqrt{t}} \exp\left(-\frac{1}{2}\frac{(\mathrm{x}-\mathrm{y})^2}{t\sigma^2}\right) V(\mathrm{y}) d\mathrm{y} \\ &= \frac{\sigma^2}{2} \frac{\partial^2}{\partial \mathrm{x}^2} \mathscr{B}^{t\sigma^2} V(\mathrm{x}) \end{aligned} \tag{3.29}$$

Examples The $\mathscr{B}$ operator is difficult to solve analytically. Only very few examples can be derived from (3.28) and expressed in closed form.

$$\mathscr{B}_{\mathrm{x}}^{\sigma^2} \exp(\mathrm{x}) = \exp\left(\mathrm{x} + \frac{\sigma^2}{2}\right) \tag{3.30}$$

$$\mathscr{B}_{\mathrm{x}}^{\sigma^2} a + b\mathrm{x} + c\mathrm{x}^2 = a + c\sigma^2 + b\mathrm{x} + c\mathrm{x}^2 \tag{3.31}$$

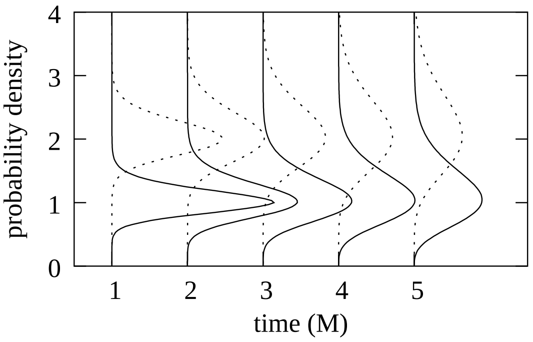


Figure 17: Geometric Brownian motion applied to a delta function.

Plotted function:
$f(M, \mathrm{x}) = \Theta^M V(\mathrm{x})$
$\Theta = \mathcal{B}^{(\sigma \mathrm{x})^2}$, with $\sigma = 20\%$
$V_1(\mathrm{x}) = \delta(1 - \mathrm{x})$,
$V_2(\mathrm{x}) = \delta(2 - \mathrm{x})$

Geometric Brownian motion

The volatility of many economic parameters is often observed to be more or less proportional to their absolute level. Random innovations to the stock prices are often assumed to have a standard deviation that is proportional to the stock price itself. The operator $\mathcal{B}^{(\sigma \mathrm{x})^2}$ models a state parameter x with relative volatility σ. For easy computation it can be reduced to a blur operator with real exponent on a transformed domain.

$$\mathcal{B}_{\mathrm{x}}^{(\sigma \mathrm{x})^2} = \mathcal{T}_{\mathrm{x}}^{-\mathrm{x}(\mathrm{x}' + \frac{1}{2}\sigma^2)} \mathcal{B}_{\mathrm{x}'}^{\sigma^2} \mathcal{T}_{\mathrm{x}}^{\mathrm{x}\mathrm{x}'} \qquad \text{for } \sigma \in \mathbb{R}_+ \tag{3.32}$$

The application to a function $V(\mathrm{x})$ can be evaluated explicitly. The proof can be found below.

$$\begin{aligned}
\mathcal{B}^{(\sigma \mathrm{x})^2} V(\mathrm{x}) &\overset{(3.32)}{=} \mathcal{T}_{\mathrm{x}}^{-\mathrm{x}(\mathrm{x}' + \frac{1}{2}\sigma^2)} \mathcal{B}_{\mathrm{x}'}^{\sigma^2} \mathcal{T}_{\mathrm{x}}^{x\mathrm{x}'} V(\mathrm{x}) \qquad (3.33)\\
&\overset{(3.21)}{=} \mathcal{T}_{\mathrm{x}}^{-\mathrm{x}(\mathrm{x}' + \frac{1}{2}\sigma^2)} \mathcal{B}_{\mathrm{x}'}^{\sigma^2} V(\mathrm{x}e^{\mathrm{x}'})\\
&\overset{(3.28)}{=} \mathcal{T}_{\mathrm{x}}^{-\mathrm{x}(\mathrm{x}' + \frac{1}{2}\sigma^2)} \frac{1}{\sigma\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left(-\frac{1}{2}\left(\frac{\mathrm{x}' - \mathrm{y}}{\sigma}\right)^2\right) V(\mathrm{x}e^{\mathrm{y}}) d\mathrm{y}\\
&\overset{(3.21)}{=} \frac{1}{\sigma\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left(-\frac{1}{2}\left(\frac{\mathrm{x}' - \mathrm{y}}{\sigma}\right)^2\right) V(\mathrm{x}e^{\mathrm{y} - \mathrm{x}' - \frac{1}{2}\sigma^2}) d\mathrm{y}\\
&= \frac{1}{\sigma\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left(-\frac{u^2}{2\sigma^2}\right) V(\mathrm{x}e^{u - \frac{1}{2}\sigma^2}) du
\end{aligned}$$

Proof: The operator terms in (3.32) can be shown to solve the same differential equation for the exponent $\sigma^2 t$.

$$\begin{aligned}
& \frac{\partial}{\partial t}\mathcal{T}_{\mathrm{x}}^{-\mathrm{x}(\mathrm{x}'+\frac{1}{2}\sigma^2)}\,\mathcal{B}_{\mathrm{x}'}^{\sigma^2 t}\,\mathcal{T}_{\mathrm{x}}^{\mathrm{x}\mathrm{x}'}V(\mathrm{x}) && (3.34)\\
\overset{(3.33)}{=}\quad & \frac{\partial}{\partial t}\frac{1}{\sigma\sqrt{2\pi t}}\int_{\mathbb{R}}\exp\left(-\frac{u^2}{2\sigma^2 t}\right)V(\mathrm{x}e^{u-\frac{1}{2}\sigma^2 t})du \\
& \text{let } \mathrm{y}{=}u{-}\ln(\mathrm{x}){+}\tfrac{1}{2}\sigma^2 \\
=\quad & \frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}}\frac{\partial}{\partial t}\underbrace{\frac{\exp\left(-\frac{\left(\mathrm{y}+\ln(\mathrm{x})-\frac{1}{2}\sigma^2 t\right)^2}{2\sigma^2 t}\right)}{\sigma\sqrt{t}}}_{=:g}V(e^{\mathrm{y}})d\mathrm{y} \\
=\quad & \frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}}\left(\frac{(\mathrm{y}-\ln(\mathrm{x}))^2}{2\sigma^2 t^2}-\frac{1}{2t}-\frac{\sigma^2}{8}\right)gV(e^{\mathrm{y}})d\mathrm{y} \\
=\quad & \mathrm{x}^2\frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}}\frac{\partial}{\partial \mathrm{x}}\left(\frac{-\mathrm{y}-\ln(\mathrm{x})+\frac{\sigma^2 t}{2}}{2t\mathrm{x}}\right)gV(e^{\mathrm{y}})d\mathrm{y} \\
=\quad & \frac{(\mathrm{x}\sigma)^2 t}{2}\frac{\partial^2}{\partial \mathrm{x}^2}\mathcal{B}^{(\mathrm{x}\sigma)^2 t}V(\mathrm{x})
\end{aligned}$$

Examples: The following result allows us to solve the geometric Brownian motion for all polynomials.

$$\mathcal{B}^{(\mathrm{x}\sigma)^2}\mathrm{x}^n = \mathrm{x}^n\exp\left(\frac{n(n-1)}{2}\sigma^2\right), \qquad \forall n\in\mathbb{R} \tag{3.35}$$

The analytic derivation is easy to verify with (3.26):
$\frac{\partial}{\partial t}\mathcal{B}^{(\mathrm{x}\sigma)^2 t}\mathrm{x}^n = \frac{\partial}{\partial t}\mathrm{x}^n e^{\frac{n(n-1)}{2}\sigma^2 t} = \frac{\sigma^2\mathrm{x}^2}{2}\frac{\partial^2}{\partial x^2}\mathrm{x}^n e^{\frac{n(n-1)}{2}\sigma^2 t} = \frac{\sigma^2\mathrm{x}^2}{2}\frac{\partial^2}{\partial x^2}\mathcal{B}^{(\mathrm{x}\sigma)^2 t}\mathrm{x}^n$.

The computation for the exponential function via a power series fails due to diverging coefficients. For positive x the series goes to ∞.

$$\mathcal{B}^{(\mathrm{x}\sigma)^2}\exp(\mathrm{x}) = \infty \qquad \forall\,\mathrm{x},\sigma>0 \tag{3.36}$$

Following (3.32) we can conclude the result for the logarithmic operand.

$$\begin{aligned}
\mathcal{B}^{(\mathrm{x}\sigma)^2}\ln(\mathrm{x}) \quad & \overset{(3.32)}{=} \quad \mathcal{T}_{\mathrm{x}}^{-\mathrm{x}(\mathrm{x}'+\frac{1}{2}\sigma^2)}\mathcal{B}_{\mathrm{x}'}^{\sigma^2}\mathcal{T}_{\mathrm{x}}^{\mathrm{x}\mathrm{x}'}\ln(\mathrm{x}) && (3.37)\\
& \overset{(3.21)}{=} \quad \mathcal{T}_{\mathrm{x}}^{-\mathrm{x}(\mathrm{x}'+\frac{1}{2}\sigma^2)}\mathcal{B}_{\mathrm{x}'}^{\sigma^2}\ln(\mathrm{x}e^{\mathrm{x}'}) \\
& \overset{(3.31)}{=} \quad \mathcal{T}_{\mathrm{x}}^{-\mathrm{x}(\mathrm{x}'+\frac{1}{2}\sigma^2)}\ln(\mathrm{x})+\mathrm{x}' \\
& = \quad \mathcal{T}_{\mathrm{x}}^{-\mathrm{x}(\mathrm{x}'+\frac{1}{2}\sigma^2)}\ln(\mathrm{x}e^{-(\mathrm{x}'+\frac{1}{2}\sigma^2)})+\mathrm{x}' \\
& = \quad \ln(\mathrm{x})-\frac{\sigma^2}{2}
\end{aligned}$$

Principal axis transformation

The multi-dimensional heat equation with constant coefficients is commonly solved on a transformed domain. It is possible to separate the individual diffusions into a sequence of one-dimensional blur operators. We assume that the covariance matrix Σ is real and independent of the diffusion directions x_i. Furthermore, the matrix must always be positive semidefinite and thus have a Cholesky decomposition Q.

$$\Sigma = QQ^{\mathsf{T}}, \quad \Sigma \in \mathbb{R}^{n \times n} \tag{3.38}$$

The domain transformation is performed by a forward and a backward transformation operator F and F^{-1} applied before and after the diffusion. The blur sequence is applied on the temporary transformation variables x_i'. We let the transformation operator F be defined as a sequence of drift operators.

$$\begin{aligned} F &= \bigodot_{i=1}^{n} \mathcal{T}_{\mathrm{x}_i}^{-e_i^{\mathsf{T}} Q \mathrm{x}'} = \mathcal{T}_{\mathrm{x}_1}^{-e_1^{\mathsf{T}} Q \mathrm{x}'} \cdots \mathcal{T}_{\mathrm{x}_n}^{-e_n^{\mathsf{T}} Q \mathrm{x}'} \\ F^{-1} &= \bigodot_{i=1}^{n} \mathcal{T}_{\mathrm{x}_i}^{e_i^{\mathsf{T}} Q \mathrm{x}'} = \mathcal{T}_{\mathrm{x}_1}^{e_1^{\mathsf{T}} Q \mathrm{x}'} \cdots \mathcal{T}_{\mathrm{x}_n}^{e_n^{\mathsf{T}} Q \mathrm{x}'} \end{aligned} \tag{3.39}$$

Lemma: The multidimensional blur $\mathcal{B}^{\Sigma}$ with covariance matrix Σ is easily evaluated by a sequence of univariate blurs and the transformation F.

$$\begin{aligned} \mathcal{B}^{\Sigma} &= F \left(\mathcal{B}_{\mathrm{x}_1'} \cdots \mathcal{B}_{\mathrm{x}_n'} \right) F^{-1} \\ &= F \left(\bigodot_{i=1}^{n} \mathcal{B}_{\mathrm{x}_i'} \right) F^{-1} \end{aligned} \tag{3.40}$$

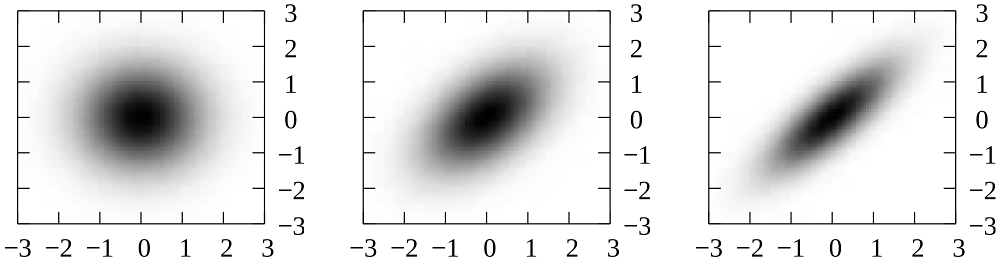


Figure 18: Joint processes with correlations 0, 0.5 and 0.8. All three processes have equal marginal distributions. The process operators are applied to a delta function located at the origin.

Plotted functions: $\mathcal{B}^{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}} \delta(\mathrm{x})$, $\mathcal{B}^{\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}} \delta(\mathrm{x})$, $\mathcal{B}^{\begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}} \delta(\mathrm{x})$

Proof: Since all operators have been solved previously, the correctness of the domain transformation (3.40) can be verified by consecutive application of each operator.

$$\begin{aligned}
& \frac{\partial}{\partial t}\mathcal{T}_{\mathrm{x}_1}^{-e_1Q\mathrm{x}'}\mathcal{T}_{\mathrm{x}_2}^{-e_2Q\mathrm{x}'}\mathcal{B}_{\mathrm{x}_1'}^{t}\mathcal{B}_{\mathrm{x}_2'}^{t}\mathcal{T}_{\mathrm{x}_1}^{e_1Q\mathrm{x}'}\mathcal{T}_{\mathrm{x}_2}^{e_2Q\mathrm{x}'}V(\mathrm{x}_1,\mathrm{x}_2) \qquad (3.41)\\
\overset{(3.20)}{=} & \frac{\partial}{\partial t}\mathcal{T}_{\mathrm{x}_1}^{-e_1Q\mathrm{x}'}\mathcal{T}_{\mathrm{x}_2}^{-e_2Q\mathrm{x}'}\mathcal{B}_{\mathrm{x}_1'}^{t}\mathcal{B}_{\mathrm{x}_2'}^{t}V(\mathrm{x}_1+q_{11}\mathrm{x}_1'+q_{12}\mathrm{x}_2',\mathrm{x}_2+q_{21}\mathrm{x}_1'+q_{22}\mathrm{x}_2')\\
\overset{(3.26)}{=} & \mathcal{T}_{\mathrm{x}_1}^{-e_1Q\mathrm{x}'}\mathcal{T}_{\mathrm{x}_2}^{-e_2Q\mathrm{x}'}\Big[\\
& \frac{1}{2}(q_{11}^2+q_{12}^2)V_{11}+(q_{11}q_{21}+q_{12}q_{22})V_{12}+\frac{1}{2}(q_{21}^2+q_{22}^2)V_{22}\Big]\\
\overset{(3.20)}{=} & \frac{1}{2}\Sigma_{11}\frac{\partial^2}{\partial\mathrm{x}_1^2}V(\mathrm{x}_1,\mathrm{x}_2)+\Sigma_{12}\frac{\partial^2}{\partial\mathrm{x}_1 d\mathrm{x}_2}V(\mathrm{x}_1,\mathrm{x}_2)+\frac{1}{2}\Sigma_{22}\frac{\partial^2}{\partial\mathrm{x}_2^2}V(\mathrm{x}_1,\mathrm{x}_2)\\
= & \frac{1}{2}\nabla^{\mathsf{T}}\Sigma\nabla(\mathcal{B}^{\Sigma})^t
\end{aligned}$$

The final result does no longer depend on the temporary variables x_i'.

3.3 Stock price models

Now it is time to synthesize some common stochastic processes from previously derived operators. We will use the drift and the blur operator to describe Brownian motion based processes with drift. Also known as Wiener or Gaussian processes they have been used for modelling asset prices since 1900 by the French bond trader Bachelier [Bac00]. In 1965 the first stock price model was based on geometric Brownian motion by Samuelson [Sam65]. This was a significant improvement, since stock prices were ensured to stay positive. Until then all models required an expected drift rate for stock prices which by nature could not be observed directly. Possibly the greatest contribution to quantitative finance was introduced by Black and Scholes in 1973 [BS73], when they discovered that the final value of a stock option can be reproduced by a dynamic hedging strategy without risk and with costs that are independent of the expected growth rate of stocks.

3.3.1 The Black&Scholes model

Black and Scholes found the formula for the arbitrage-free option price, as the expected value of a modified process. They used a constant coefficient Brownian motion with a drift equal to the riskless interest rate and measured the expected pay off. This approach was later called risk-neutral evaluation by Cox and Ross [CR76], since stock and bond rates can only be equal if investors do not demand an extra premium for the high risks inherent in stock markets. Using the shift and blur operator, the time process Θ_{bs} of the famous model can be rewritten.

$$\Theta_{bs} = e^{-r}\mathcal{T}_S^{rS}\mathcal{B}_S^{(\sigma S)^2} \qquad r,\sigma\in\mathbb{R} \tag{3.42}$$

The process is built from three **individual operations**: discounting, stock price drift and stock price diffusion. The discount effect renders the present value of the received cash in one period (e^{-r}). The expected drift in the stock price is the interest rate ($\mathcal{T}^{rS}$). And, finally, there is uncertainty added by the diffusion operator ($\mathcal{B}^{(\sigma S)^2}$), whereas g denotes the standard normal density function.

$$\begin{aligned} \Theta_{bs} V(S) &\overset{(3.42)}{=} e^{-r}\, \mathcal{T}_S^{rS}\, \mathcal{B}_S^{(\sigma S)^2} V(S) \\ &\overset{(3.33)}{=} e^{-r}\, \mathcal{T}^{rS} \int_{\mathbb{R}} g(u) V(S e^{-\sigma u - \frac{1}{2}\sigma^2}) du \\ &\overset{(3.21)}{=} e^{-r} \int_{\mathbb{R}} g(u) V(S e^{r - \sigma u - \frac{1}{2}\sigma^2}) du \end{aligned} \tag{3.43}$$

It is easily verified that the three operators commute and can be written in any of six possible orders. Multiple applications of Θ are therefore most easily evaluated by powering each individual operator.

$$\begin{aligned} \Theta_{bs}^{\ M} V(S) &= e^{-rM}\, \mathcal{T}_S^{rSM}\, \mathcal{B}_S^{(\sigma S)^2 M} V(S) \\ &= e^{-rM} \int_{\mathbb{R}} g(u) V(S e^{rM - \sigma u M - \frac{1}{2}\sigma^2 M}) du \end{aligned} \tag{3.44}$$

Option price An option's pay off at maturity time depends on the final value of the stock price. We consider European call and put options with strike K. Their value at maturity time is V_{call} and V_{put} respectively.

$$\begin{aligned} V_{call}(S) &= \max(S - K, 0) \\ V_{put}(S) &= \max(K - S, 0) \end{aligned} \tag{3.45}$$

The expected value of the option with time to maturity M is computed by applying our Θ operator M times to the value function V. The resulting function is supplied with the current stock price S_0.

$$\underset{S=S_0}{\text{大}}\ \Theta_{bs}^{\ M}\, V \tag{3.46}$$

3.3.2 GARCH

The GARCH[2] model is a stochastic volatility model that deserves some extra attention. This is a path dependent process, where future probabilities depend on previous states. The volatility parameter σ varies with time and is driven by variations in the stock price. Any innovation to the price is fed back into the volatility parameter. The model in this section is known as GARCH(1,1) in the literature [Bol86].

$$\sigma_t^2 = (1 - \alpha - \beta)\sigma_{t-1}^2 + \alpha \ln\left(\frac{S_t}{S_{t-1}}\right)^2 + \beta\gamma \tag{3.47}$$

The process operator of the GARCH model consists of three steps. First, the current stock price S is stored in a temporary variable S'. Then any other time process, e.g. a standard Black&Scholes step, with stock price drift and diffusion is performed. Finally, the variance σ^2 is updated based on the ratio of realized stock price S and its previously expected value S'.

---

[2] Generalized AutoRegressive Conditional Heteroskedasticity

$$\Theta_{garch} = \underset{S'=\Theta_{bs}S}{大} \Theta_{bs} \underset{\substack{\sigma^2=(1-\alpha-\beta)\sigma^2+\\ \alpha\ln\left(\frac{S}{S'}\right)^2+\beta\gamma}}{大} \tag{3.48}$$

Three parameters specify the GARCH model, the update intensity α, the mean reversion speed β and the long term mean γ. Figure 19 shows the convolution kernel for $\alpha = 0.02$ and $\beta = 0$.

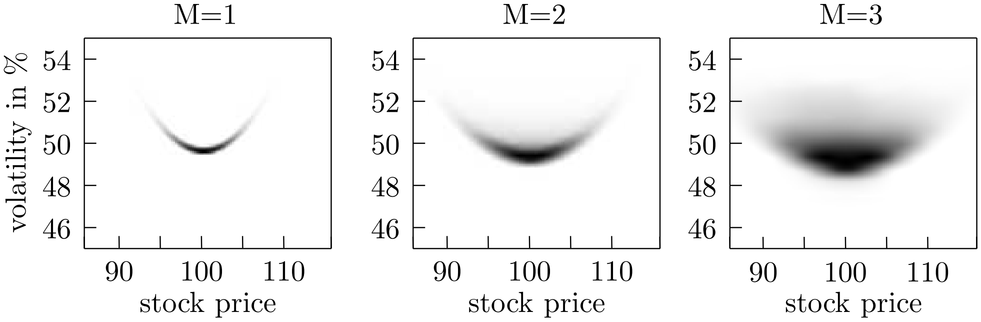


Figure 19: Joint probability distribution of volatility and stock price under the GARCH process. Large jumps of the stock price, starting at 100, coincide with a rise in volatility. Over longer time intervals this dependence fades away. The volatility becomes less dependent on the current price, but more on the path along which it got there.

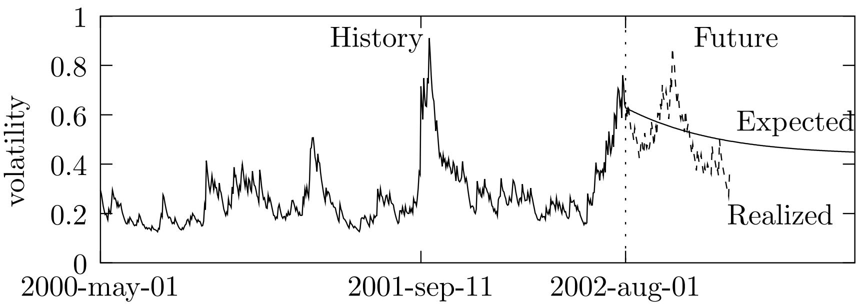


Figure 20: Volatility analysis of the European banking index (SX7E) based on August 1st, 2002. The future expectation under the GARCH model shows the mean reversion inherent in this model.

3.3.3 Copulas

So far we have looked at single asset models. For additional stocks with different sources of randomness there needs to be a definition of their interdependence structure. One standard tool was the previously derived correlation method. It can be applied easily via a linear domain transformation. In more general settings one often wishes to separate the process of individual parameters from their interdependence structure [QMRLUF03]. Then the complex interactions of various marginal processes can be modelled by a copula function. The marginal distributions Φ_i of the individual parameters x_i can be derived from a common process Θ or an individual process Θ_i.

$$\Phi_i(u) = \Theta_i \, 1_{u>\mathrm{x}_i} \tag{3.49}$$

Their statistical interdependency can be written in a very general form as a copula. The copula is the probability distribution of the uniformly distributed variables $\Phi_i(\mathrm{x}_i)$. The full process can be written in 2D as a multidimensional integral. The integral over the function V on the unit cube is equivalent to the convolution formula (3.4), but adds the interdependence structure to the interdependent variables.

$$\Theta_{cop} V(\mathrm{x}) = \int_0^1 \int_0^1 \left(\frac{\partial}{\partial u} \frac{\partial}{\partial v} C(u,v) \right) V\big(\mathrm{x}_1 - \Phi_1^{-1}(u), \mathrm{x}_2 - \Phi_2^{-1}(v)\big) du \, dv \tag{3.50}$$

A fairly general group of Archimedean copula functions is created via a probability distribution F. Valid copulas are created for all choices of continuous distributions F [QMRLUF03].

$$C_{Arch}(u,v) = F\big(F^{-1}(u) + F^{-1}(v)\big) \tag{3.51}$$

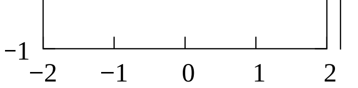


Figure 21: Convolution kernel generated by a Gaussian distribution (G) and a Poisson marginal distribution, combined with the Gumbel copula, a special case of Archimedean copula.

$\Phi_1(\mathrm{x}_1) = G(\mathrm{x}_1)$
$\Phi_2(\mathrm{x}_2) = 1 - \exp(-\mathrm{x}_2)$
$F(\mathrm{x}) = 1 - \exp\left(-\mathrm{x}^{1/\beta}\right)$
$\beta = 1/2$

Plotted function:
$p(\mathrm{x}_1', \mathrm{x}_2') = \underset{\substack{x_1=0\\x_2=0}}{大} \Theta_{cop}\, \delta(\mathrm{x}_1 - \mathrm{x}_1')\delta(\mathrm{x}_2 - \mathrm{x}_2')$

3.4 Term structure models

In this section we will explore processes with functions as state variables. In particular we investigate interest rate and yield curve models, determining the performance of riskless investments, like bonds. In risk neutral pricing these yield curves resemble the expected growth rate of all other investments.

Numéraire Before we can take a closer look at interest rate models we need to define some vocabulary. The numéraire is a unit of wealth proportional to which all cash flows are measured. We will use the symbol P_0 for one unit of cash on a money market account. Hence, P_0 is the inverse of the numéraire and measures today's present value and becomes smaller the further we look into the future.

$$P_0 := \text{unit of wealth} \tag{3.52}$$

Yield The yield Y defines the expected discount rate over the next N periods. It is is computed from the ratio of the expected numéraire value in N periods $\Theta^N P_0$ and today's value 太$P_0 = 1$. This ratio may depend on other process parameters.

$$Y(N) := -\frac{1}{N} \ln \left(\Theta^N P_0\right) \tag{3.53}$$

Forward rate Most often we will use the instantaneous forward rate $f(t,s)$, often referred to as instantaneous discount rate for time s. This curve is measured at time t.

$$f(t,s) := \frac{\partial}{\partial s}(s-t)\,Y(s-t) \tag{3.54}$$

Its initial value $f(0,s)$ is commonly considered as a known input parameter. It is the forward rate curve $f(t,s)$ supplied with all initial values.

$$f(0,s) = 太 f(t,s) \tag{3.55}$$

Drift condition Following the previous two statements, the expected value of P_0 in s periods can be derived. It is the current value P_0 multiplied by the accrued interest rate. All interest rate models must obey this condition for a given term structure $f(t,s)$.

$$\Theta^s P_0 = P_0 \exp\left(-\int_t^{t+s} f(0,u)du\right) \tag{3.56}$$

Note, the drift condition only holds in the risk-neutral world or the risk-neutral measure. Thus, the drift condition is only relevant for models that are used for pricing interest derivatives.

3.4.1 Deterministic model

In the first model we will consider interest rates to be deterministic and compute the result of contingent future cash flows. We will compare two different versions of the solution. In the first, the numéraire P_0 creates discounted cash flows and in the second accrues cash accounts and returns expected final values.

Version 1: The first version is probably preferred in practice, since it directly results in discounted present values. The deterministic term structure model consists of two effects. First, the numéraire value is discounted with the forward rate of the next period. Second the time counter t is updated.

$$\Theta^{\Delta t}_{det} = \mathcal{T}_{P_0}^{-P_0 \int_t^{t+\Delta t} f(0,u)du} \mathcal{T}_t^{\Delta t} \tag{3.57}$$

In a simple example we compute the present value of an annuity that pays one unit of wealth per period. Thus, we evaluate a repeated operator sequence that waits one unit of time with Θ and then transfers one unit of cash onto account c. Inserting the constant forward curve, $f(0,s) = r = 0.1$, our initial wealth c_0 increases by $6.01P_0$.

$$\begin{aligned}
大\left(\Theta_{det}\mathcal{T}_c^{1P_0}\right)^{10} c &= 大\left(\Theta_{det}\mathcal{T}_c^{1P_0}\right)^{9} \mathcal{T}_{P_0}^{-rP_0}(c + 1P_0) \qquad (3.58)\\
&= 大\left(\Theta_{det}\mathcal{T}_c^{1P_0}\right)^{9} (c + e^{-r}P_0)\\
&= 大\left(\Theta_{det}\mathcal{T}_c^{1P_0}\right)^{8} \mathcal{T}_{P_0}^{-rP_0}(c + P_0 + e^{-r}P_0)\\
&= 大\left(\Theta_{det}\mathcal{T}_c^{1P_0}\right)^{8} (c + e^{-r}P_0 + e^{-2r}P_0)\\
&= \cdots\\
&= c_0 + \sum_{t=1}^{10} e^{-0.1t}P_0\\
&= c_0 + 6.01041P_0
\end{aligned}$$

It is important to realize, that operator sequences are always written chronologically in time. First, we wait, then we pay and repeat both effects ten times.

Version 2: The second version is sometimes easier to write, but discounts cash flows to the end of the model horizon. We increase the cash account c in every period. Thus our time operator Θ pays the coupon and updates time t.

$$\Theta^{\Delta t}_{fut} = \mathcal{T}_c^{c\int_t^{t+\Delta t} f(u)du} \mathcal{T}_t^{\Delta t} \tag{3.59}$$

The same example as above results in the cash amount that we will possess after the ten periods passed. Our initial value c_0 and the return from the investment is larger by e^1.

$$大\left(\Theta_{fut}\mathcal{T}_c^{1}\right)^{10} c = 16.338 + 2.71828c_0 \tag{3.60}$$

3.4.2 Heath-Jarrow-Morton

The Heath-Jarrow-Morton model is one of the most general term structure models. It describes the stochastic behavior of the whole forward rate curve f in a risk neutral world and is suited for the evaluation of interest rate derivatives. In discretized time the HJM model updates the curve $f(t,s)$ with a deterministic drift $\mu(t,s)$ and normally distributed impact with deterministic volatility $\sigma(t,s)$ [Mun98].

$$f(t + \Delta t, s) = f(t,s) + \sigma(t,s)\xi\sqrt{\Delta t} + \mu(t,s)\Delta t, \quad \xi \sim \mathcal{N}(0,1) \tag{3.61}$$

Since the process models the evolution of $f(t,s)$ for all s simultaneously we need to find a way to apply our differential operators to curves, or in particular to express the curve in terms of finitely many coefficients. We define the current forward rate curve $f(t,s)$ as the sum of the initial curve $f(0,s)$ and a linear combination of a set of basis functions ϕ_i. The linear coefficients a_i are initialized to zero and vary with time. Instead of modeling curves the HJM model describes the evolution of the coefficients a_i.

$$f(t,s) = f(0,s) + \sum_{i=0}^{N} a_i \phi_i(s) \tag{3.62}$$

For the sake of simplicity, we assume that ϕ_i is an orthonormal basis with respect to an inner product A.

$$\begin{aligned} < \phi_i | \phi_j > &= \int_0^\infty \phi_i(s) A(s) \phi_j(s) ds \\ &= \delta_{ij} \end{aligned} \tag{3.63}$$

Although the concept of basis functions was not explicitly part of the original work on the HJM model [Mun98, HJM92], it is the common numerical procedure for modeling curve transformations. The HJM model now consists of a diffusion $\mathcal{B}$ and convection $\mathcal{T}$ in the coefficient vector a. After the curve is updated, the process continues as in the deterministic model, paying a coupon and updating the time counter (3.57).

$$\Theta_{hjm}^{\Delta t} = \mathcal{B}_a^{\Sigma \Delta t} \, \mathcal{T}_a^{R \Delta t} \, \Theta_{det}^{\Delta t} \tag{3.64}$$

Due to the orthogonal basis functions, the variance matrix is diagonal. The drift vector s projects a drift μ onto the function basis ϕ.

$$\Sigma = \begin{pmatrix} < \sigma | \phi_1 >^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & < \sigma | \phi_N >^2 \end{pmatrix}, \quad R = \begin{pmatrix} < \mu | \phi_1 > \\ \vdots \\ < \mu | \phi_N > \end{pmatrix} \tag{3.65}$$

The drift μ that satisfies the initial condition (3.56) has been derived multiple times by various sources [HJM92, Mun98, Shr97, Zag02].

$$\mu := \sigma(t,s) \int_t^s \sigma(t,u) \, du. \tag{3.66}$$

It has to be noted, that the risk neutral drift μ was derived in continuous time, whereas Θ-processes always operate in discrete time. The result is correct for small Δt and volatilities σ that do not depend on f.

3.4.3 Hull&White

There exists a range of volatility structures σ for which the drift adjustments μ can be captured exactly with a finite set of basis functions ϕ_i [IK98]. The Hull and White interest rate model is a special case of the extended Vasicek model and requires only two basis functions for its oscillation on the initial curve. It can also be written as a special case of the HJM model [Zag02, p134] with an exponentially decaying σ with the volatility level α and the volatility decay κ that makes longer forward rates less volatile.

$$\sigma = \alpha \exp\big(-\kappa(s-t)\big) \qquad \alpha,\kappa \in \mathbb{R}_+ \tag{3.67}$$

In order to represent shifts in the interest rate curve $f(t,s)$ together with the drift adjustment term μ, we need the function space spanned by σ and $-\sigma \int \sigma ds$. These can be represented in an orthonormal basis ϕ_1 and ϕ_2, with respect to the standard inner product ($A = 1$).

$$\begin{aligned} \phi_1(s) &= \sqrt{2\kappa}\exp(-\kappa s) \\ \phi_2(s) &= \sqrt{36\kappa}\left(\exp(-2\kappa s) - \frac{\sqrt{2}}{3\sqrt{\kappa}}\,\phi_1(s)\right) \end{aligned} \tag{3.68}$$

For this function basis the results of the inner products can be given explicitly. These parameters determine the HJM model.

$$\begin{aligned} <\sigma|\phi_1> &= \frac{\alpha\exp(\kappa t)}{\sqrt{2\kappa}} \\ <\sigma|\phi_2> &= 0 \\ <\mu|\phi_1> &= -\frac{\alpha^2\sqrt{2}\exp(\kappa t)(2\exp(\kappa t)-3)}{6\kappa^{\frac{3}{2}}} \\ <\mu|\phi_2> &= -\frac{\alpha^2\exp(2\kappa t)}{6\kappa^{\frac{3}{2}}} \end{aligned} \tag{3.69}$$

The basis functions ϕ_1 and ϕ_2 may not depend on current time t, while the inner products may not depend on maturity s. If that happens to be the case, you might have chosen an inappropriate function basis. The Hull and White model is written according to (3.64). Since σ and μ are independent of a we can expand the operators into a sequence of univariate operations.

$$\Theta_{hw} = \mathcal{B}_{a_1}^{<\sigma|\phi_1>}\mathcal{T}_{a_1}^{<\mu|\phi_1>}\mathcal{T}_{a_2}^{<\mu|\phi_2>}\Theta_{det}^1 \tag{3.70}$$

The sequence consists consists of the usual effects. A diffusion and drift in the curve parameters a and the effects from the deterministic model. The operator Θ_{hw} can be raised to the power of Δt, which is approximated by powering each individual operator. Since the operator term denotes a discrete time version of the model the accuracy depends the time resolution Δt.

Example: As an example for the Hull&White interest rate model we will evaluate a bond deal. Suppose at time zero you paid 0.8 units of wealth and receive a redemption of one unit after one period. The full operator sequence for this deal is denoted by Π and consists of three effects. In chronological order you pay 0.8, then wait Θ and finally add one to the account c.

$$\Pi(\Theta) := \mathcal{T}_c^{-0.8P_0}\,\Theta\,\mathcal{T}_c^{1P_0} \tag{3.71}$$

The result of the operator sequence can be solved analytically and numerically for quite a range of operand functions. We assume values for f, α and κ.

$$\begin{aligned} f(0,s) &= 0.1 + 0.01s && (3.72)\\ \alpha &= 0.2 \\ \kappa &= 0.5 \\ c_0 &= 0 && (3.73) \end{aligned}$$

Hence we have a formula for $f(t,s)$ that depends only on the two time dependent variables a_1 and a_2.

$$f(t,s) = a_1 \exp(-0.5s) + a_2\sqrt{18}\left(\exp(-s) - \frac{2}{3}\exp(-0.5s)\right) \tag{3.74}$$

The cash account c starts at zero. This value must be inserted into the function, after the operator $\Pi(\Theta)$ was applied. Before application c has to be treated as either a free variable or as a free dimension of a multidimensional function. According to (3.3) the expected discounted profit of this investment is $\Pi(\Theta)c$.

$$\mathbb{E}(c) = \text{大}\,\Pi(\Theta_{hw})\,c = 0.100P_0 \tag{3.75}$$

The application of univariate blur operators has been discussed mulitple times and expands to extremely lengthy algebraic terms. For that reasons we display only the numeric results.
Following the same rationale we can compute the expected squared profit.

$$\mathbb{E}(c^2) = \text{大}\,\Pi(\Theta_{hw})\,c^2 = 0.0415P_0^2 \tag{3.76}$$

Having expectations for c and c^2 we can compute the standard deviation of discounted profits.

$$\sqrt{\mathbb{E}(c^2) - (\mathbb{E}c)^2} = 0.177P_0 \tag{3.77}$$

The operator $\Pi(\Theta)$ can be supplied with any measuring function and we could extract all kinds of risk measures and expected values for the investment Π under the process Θ_{hw}.

3.5 Example

This final example demonstrates the flexibility of the Θ-notation for stochastic processes. Based on a Θ operator that models economic parameters we can derive explicit formulas for the evaluation of various trading strategies and financial portfolios. Applying the product operator to appropriate measuring functions will reveal all kinds of risk measures and expected values.
Suppose you entered or are offered to enter a financial contract of the following type: A bond issuer wants to borrow from you a certain amount of money and in turn binds himself to a dept redemption procedure. For a period of 10 years he pays a coupon of 7 cash units per year and finally redeems a nominal value of 100. Furthermore the debtor has the option of early redemption at a strike

price of 105. The option can be exercised right after each coupon payment, but not before year two. The investment can be written as an operator $\Pi(\Theta)$ and reads chronologically from left to right. The Θ refers to a unit period of inactivity, $\mathcal{T}_c$ transfers the exponents amount onto the cash account and min selects the less valuable of two functions.

$$\Pi(\Theta)V = \left(\Theta\mathcal{T}_c^{7P_0}\right)^2 \left(\begin{array}{l} \overset{\min}{\nearrow}\ \mathcal{T}_c^{105P_0}V \\ \searrow\ . \end{array} \quad \Theta\mathcal{T}_c^{7P_0} \right)^8 \mathcal{T}_c^{100P_0}\,V \tag{3.78}$$

When entered into a computer algebra system, the term is flattened out and expanded into nested minimum functions. The algebraic result can become very long and should not necessarily be looked at.

For our numeric examples we choose a Hull&White model with an additional probability for the debtors default λ. In case the debtor goes default the portfolio operator $\Pi(\Theta)$ is not continued. Instead we evaluate V directly, without any further payments. The function can be considered as a constant operator that evaluates V regardless of what operand it is applied to. Due to Θ's definition via the expected value (3.3) the two scenarios can be combined linearly (see 2.4.2).

$$\Theta = \begin{array}{l} \overset{1-\lambda}{\nearrow}\ \Theta_{hw} \\ \underset{\lambda}{\searrow}\ V \end{array} = \lambda V + (1-\lambda)\Theta_r \tag{3.79}$$

The model is parameterize by two kinds of parameters. One that are constant and could as well be given as numbers and others which behave like free variables at first and are only inserted after all operators are applied. The point of insertion has a crucial effect on the numeric result. We will use the 大 operator to indicate the replacement of variables with their initial coordinates.

$$大V = V\big|_{\substack{c=0\\t=0}} \tag{3.80}$$

Depending on the system you use there are three different implementations of the insertion procedure. Computer algebra systems substitute the numeric values for the free variables and triggers a numeric evaluation scheme. Monte Carlo methods use the initial values for scenario generation. PDE methods solve the operator term on a multidimensional grid of which each free variable refers to one grid dimension. And, finally, evaluate the grid at the initial values. The model parameters that serve just as an abbreviation of well defined values can be inserted into the function at any time. The probability measure for λ fit to the risk-neutral probabilities of Θ_{hw}.

$$\lambda = \frac{1}{100}, \quad r_0 = \frac{5}{100}, \quad f(0,s) = r_0 + \frac{s}{200}, \quad \sigma = 0.02e^{-0.66(s-t)}$$

3.5.1 Present value

The first and most commonly asked question is about the investment's present value. We can compute the expected discounted cash value by setting our measure function V to the cash account c. Accordingly, any application of the portfolio operator $\Pi(\Theta)$ will evaluate the expected final value of c under process Θ.

$$V_c(c,t) = c \tag{3.81}$$

Figure 22 shows the discounted portfolio value and its sensitivity to parallel shifts of the interest rate curve. The horizontal axis shows the interest rate at start time r_0 . On the vertical axis we can see the values of three different investments. The bond with short option PV is less valuable than the two reference investments: the same portfolio without short option, i.e. a 10 year coupon bond (BV) and the portfolio, if exercised after two periods, i.e. a two year bond with a redemption of 105 (EV).

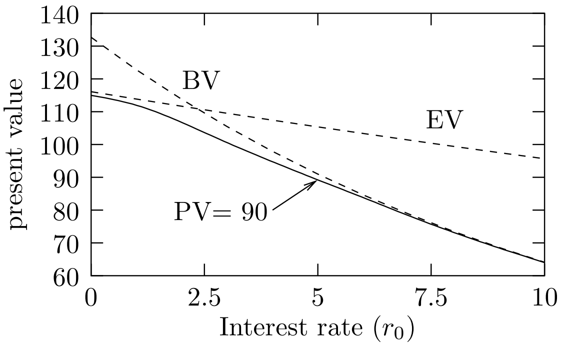


Figure 22: A bond with short call is less valuable than the plain bond and the first exercise value.
Portfolio value:
$PV(r_0) = \Pi(\Theta)V_c$
Bond value:
$BV(r_0) = \left(\Theta\mathcal{T}_c^{7P_0}\right)^{10}\mathcal{T}_c^{100P_0}V_c$
First exercise value:
$EV(r_0) = \left(\Theta\mathcal{T}_c^{7P_0}\right)^{2}\mathcal{T}_c^{105P_0}V_c$

3.5.2 Risk measures

A probability distribution for discounted portfolio values can be computed with an extended measure function $\widetilde{V}$. The new measure has an additional component containing a characteristic function on where the original measure V is smaller than a treshold x. The expected value of the indicator function, as evaluated by the portfolio operator Π equals the probability of the cash value to be less than x.

$$\widetilde{V} = \begin{pmatrix} V_c \\ 1_{V_c<\mathrm{x}} \end{pmatrix} \tag{3.82}$$

The first component is required to determine whether to exercise the option, as required by the minimum function in the portfolio operator Π. The expected value of the second component is equal to the cumulative probability distribution of discounted final profits on account c. Figure 23 plots its derivative, the probability density.

3.5.3 Stopping times

Another important question concerns the time after which the option is exercised. The portfolio is now evaluated with a new parasite component in $\widetilde{V}$. The stopping time indicator determines the probability that the option was

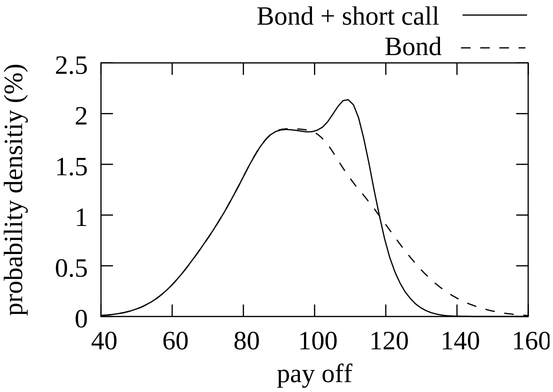


Figure 23: In case of a plain bond, the probability distribution of profits, exhibits the usual bell shape. Taking the short call into the portfolio rules out very high revenues in favor of early payment just above the strike price.

Plotted function:

$$\frac{d}{d\mathrm{x}}\Pi(\Theta)\begin{pmatrix} V_c \\ 1_{V_c<\mathrm{x}} \end{pmatrix}$$

exercised within the first τ time steps.

$$\widetilde{V} = \begin{pmatrix} V_c \\ 1_{t<\tau\wedge c\geq 105P_0} \end{pmatrix} \tag{3.83}$$

There are two possible reasons why the time variable t is smaller than maturity time. One is that the option was exercised and the portfolio was not evaluated to the end. The second explanation is a default. We are interested in option exercises and can differentiate scenarios by checking whether a cash redemption was payed. According to figure 24 the total probability of exercise is $\Pi(\Theta)\widetilde{V}_2 = 27.7$.

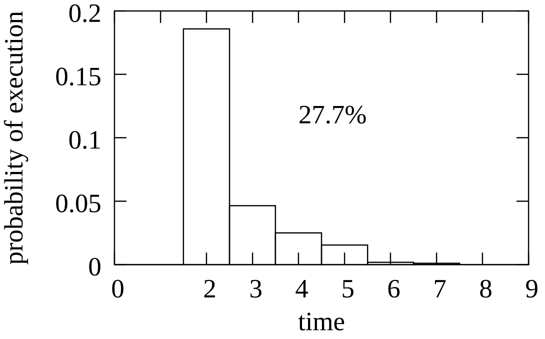


Figure 24: Probability of option exercise. The option becomes increasingly unlikely to be exercised with time. The total probability of exercise is 27.7%.

Plotted function:

$$\frac{d}{d\tau}\Pi(\Theta)\begin{pmatrix} V_c \\ 1_{t<\tau\wedge c\geq 105P_0} \end{pmatrix}$$

3.6 Conclusion

This chapter introduced a new financial calculus based on the foundations of operator theory. Any stochastic process can be written as an operator Θ, which in turn can be constructed by simple operators, that introduce various deterministic or stochastic effects. Most notably are the operators $\mathcal{T}$ and $\mathcal{B}$ that solve the transport and the heat equation. Whereas $\mathcal{T}$ models deterministic impacts and transactions between various accounting variables. Random or Gaussian impacts are performed by the $\mathcal{B}$ operator. In combination both operators allow the description of Brownian motions with drift, for processes like Black&Scholes

or interest rate models. An event sequence that occurs within one time step can be summarized in the operator Θ. Read chronologically from left to right the operators also allow the notation of trading strategies and financial portfolios (see chapter 2).

The operator notation can easily be implemented in the context of a computer algebra system or any programming environment that supports the concept of functions as data types. Hence, Θ-calculus presents an ideal tool for financial engineers who wish to quickly set up various financial models and need the freedom to combine processes, investment strategies and risk or return measures easily.

4 Put into practice

In this final chapter we discuss a simple financial product and its transformation into a numeric scheme. First, we will use the operator notation developed in chapters 2 and 3 to describe the financial contract and a model for the stock price dynamics. Then we will demonstrate that the resulting operator term is flexible enough to extract some analytic properties. Based on three different representations of the operator term there will be three possible numeric implementations, that consist of hardly more than a transcription of each individual operator in an operator sequence. This quick transition from operators to computer code is made possible by chapter 1, where an abstract class for functions and functional operators was implemented. Finally, we will show how easy it is to extend the scheme to adaptive refinement. Normally only a result of heavy coding, we will just introduce an additional operator and set up an adequate error estimator.

4.1 Economic model

This first section derives the economic setting in operator notation. There are two main advantages of operator-based finance over traditional notations. First, operators allow the explicit notation of all kinds of financial contracts, where prevailing financial literature can currently only present text form descriptions. Second, the operator notation is much closer to a computational scheme. Evaluating mathematical properties requires no more than the insertion of appropriate definitions, as it is suitable for computer algebra and numerics software.

4.1.1 The product

We will consider a product known as Asian option with strike $K = 1$ and time to maturity $n = 5$, as introduced in (2.31). The precise details of this contract can be represented by a sequence of effects in chronological order. Initially, a temporary accounting variable a is introduced and set to zero. Then follows an event sequence that is repeated five times, as indicated by the operator power. This sequence performs a time step Θ, during which economic parameters can fluctuate according to a theoretical model, and then adds the current stock price S onto the counter a. After the repeated sequence finishes, an option between nothing and a cash payment of $a/5 - 1$ is exercised.

$$\begin{tikzcd}[column sep=small] & \mathcal{T}_c^{a/5-1} \\ \Pi(\Theta) = \underset{a=0}{\text{大}} \left(\Theta \mathcal{T}_a^{S}\right)^5 \arrow[ur, no head, "\max"] \arrow[dr, no head] & \\ & \cdot \end{tikzcd} \tag{4.1}$$

The portfolio $\Pi(\Theta)$ is represented by an operator that depends on a process for economic parameters Θ. It can be applied to any measuring function, depending on what kind of statistical property we want to extract. The most common measure is the pricing measure, that evaluates the expected amount on the cash account c after contract maturity. Assuming an initial balance of $c = 0$, we can slightly simplify the previous operator term (4.1).

$$\underset{c=0}{\text{大}}\,\Pi(\Theta)\,c = \underset{a=0}{\text{大}}\,\left(\Theta\mathcal{T}_a^S\right)^5 \max(a/5-1,0) \tag{4.2}$$

4.1.2 The process

As the model for the stock price dynamics we use the Black&Scholes model as defined in (3.42). The process assumes stock prices with normally distributed movements and an expected stock price growth that is equal to the interest rate. This growth property ensures that expected cash values are actual prices that can be reproduced by a hedging strategy (see 2.5).

$$\Theta_{bs} = e^{-r}\mathcal{T}_S^{rS}\,\mathcal{B}_S^{(\sigma S)^2} \tag{4.3}$$

The interest rate r occurs twice in this equation. Once for the stock price drift and once for the discounting. The later is done by a multiplication with e^{-r}, which is only allowed if applied to a measure that is proportional to a cash amount, i.e. a measure for pricing. We assume the following values for r and σ.

$$r = \frac{2}{100}, \qquad \sigma = \frac{40}{100} \tag{4.4}$$

4.2 Algebraic results

The task of computing the option price according to (4.2) can not be performed analytically. However, we can derive other properties and preprocess the operator term for effective solution with various numerical methods. The algebraic results that are presented in this section are not new and several methods are known to derive them. What is going to be demonstrated here is that the operator notation provides quick access to these results. Thus, the notation is a compact description language for finance and its computational tasks.

4.2.1 Simplified setting

The reason why the pricing formula can not be solved algebraically is the maximum function in the final payment. It is not smooth and can not be integrated algebraically. We simplify the function (4.2) and replace it with a polynomial that depends on a.

$$\underset{a=0}{\text{大}}\,\left(\Theta\mathcal{T}_a^S\right)^5 a^p \tag{4.5}$$

With this simplified pay off there is a straight forward solution according to the previously derived algebraic rules for the involved operators $\mathcal{B}$ and $\mathcal{T}$.

Term	Result
$(\Theta \mathcal{T}_a^S)^5\, a$	$4.805S$
$(\Theta \mathcal{T}_a^S)^5\, a^2$	$37.209S^2$
$(\Theta \mathcal{T}_a^S)^5\, a^3$	$955.140S^3$

There are several reasons why one could be interested in the results of this simplified option. Firstly, it can be used as a test case for numerical option pricers. Their accuracy can be estimated based on these similar but solvable problems. Secondly, the price of options with the polynomial payoff can be used to extract the expected value and higher order moments of the final distribution of a. Analytical approximations can be built by generating a process for a that has the same moments.

4.2.2 Integral form

Financial products, including the Asian option, are often evaluated by numerical integration. A mathematical representation for the financial product must therefore be converted into integral form. Without a proper mathematical term for the product this might be an easy task for the experienced, but not trivial for anyone who is new to finance. With the explicit operator notation the task can be performed mechanically, because the operator notation is essentially an integral representation. The integral is hidden in the blur operator $\mathcal{B}$ and only needs to be expanded according to $\mathcal{B}$'s definition (3.33).

$$\begin{aligned}
\Theta \mathcal{T}_a^S f(a,S) &= e^{-r} \mathcal{T}_S^{rS}\, \mathcal{B}_S^{(\sigma S)^2}\, \mathcal{T}_a^S f(a,S) \qquad (4.6)\\
&= \frac{e^{-r}}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-1/2\, u^2}\, \mathcal{T}_S^{S(r-\sigma u-\sigma^2/2)}\, \mathcal{T}_a^S f(a,S)\, du\\
&= \frac{e^{-r}}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-1/2\, u^2} f\left(a + Se^{r-\sigma u-\sigma^2/2}, Se^{r-\sigma u-\sigma^2/2}\right) du
\end{aligned}$$

As a further preparation for numerical quadrature, the integration domain is often mapped onto a finite interval. It is suggested to substitute the integration variable u with the inverse Gaussian distribution. Thus, the integration borders are now 0 and 1, with the nice side effect that the exponential function in the integrand cancels out.

$$\Theta \mathcal{T}_a^S f(a,S) = e^{-r} \int_0^1 \mathcal{T}_S^{S\left(r-\sigma G^{-1}(v)-\sigma^2/2\right)}\, \mathcal{T}_a^S f(a,S)\, f(a,S)\, dv \qquad (4.7)$$

Whereas G is the Gaussian distribution function.

$$G(v) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{v} e^{-1/2u^2} dv \qquad (4.8)$$

The integral form for a single time step must be applied 5 times to reach the maturity of our Asian option. This leads us to 5-dimensional integral. Due to the high dimensionality the Monte-Carlo method is the rule of choice. A numerical integration based on sparse grids, a method to create regular grids in high dimensions, is also possible but does not converge as fast, since the integrand is not smooth [BD03].

4.2.3 Logarithmic scale

The Black&Scholes model postulates a geometric Brownian motion for the stock price. It is a well known fact that geometric Brownian motion can be turned into standard Brownian motion on a logarithmic scale. The standard Brownian motion is expressed by the blur operator $\mathcal{B}$ with just a real number in the exponent and is often easier to implement.
Basic ingredients for this transformation were derived in section 3.2.2. A geometric blur $\mathcal{B}^{(\sigma S)^2}$ was shown to be equivalent to a sequence of three operators: Transformation onto log scale, blurring and transforming back to standard scale. Inserting this equivalence in our pricing formula, we end up with a sequence of blurs on log scale and transactions onto counter a on standard scale. It would be nice to perform the transaction to a on log scale as well, such that no scale changes are required. Knowing the basic rules and operator properties we can evaluate this transformation purely in operator calculus.

$$\begin{aligned}
\left(\Theta \mathcal{T}_a^S\right)^5 f(a) &= \left(e^{-r} \mathcal{T}_S^{rS} \mathcal{B}_S^{(\sigma S)^2} \mathcal{T}_a^S\right)^5 f(a) \qquad (4.9)\\
&\overset{(3.32)}{=} e^{-5r} \left(\mathcal{T}_S^{rS} \mathcal{T}_S^{-S(S'+\frac{1}{2}\sigma^2)} \mathcal{B}_{S'}^{\sigma^2} \mathcal{T}_S^{SS'} \mathcal{T}_a^S\right)^5 f(a)\\
&= e^{-5r} \mathcal{T}_S^{S(r-\frac{1}{2}\sigma^2)} \left(\mathcal{B}_{S'}^{\sigma^2} \mathcal{T}_S^{SS'} \mathcal{T}_a^S \mathcal{T}_S^{S(r-S'-\frac{1}{2}\sigma^2)}\right)^4 \mathcal{B}_{S'}^{\sigma^2} \mathcal{T}_a^{Se^{S'}} f(a)\\
&= \underset{S'=0}{\text{大}}\ e^{-5r} \mathcal{T}_S^{S(r-\frac{1}{2}\sigma^2)} \left(\mathcal{B}_{S'}^{\sigma^2} \mathcal{T}_a^{Se^{S'}} \mathcal{T}_S^{S(r-\frac{1}{2}\sigma^2)}\right)^4 \mathcal{B}_{S'}^{\sigma^2} \mathcal{T}_a^{Se^{S'}} f(a)\\
&= \underset{S'=0}{\text{大}}\ e^{-5r} \left(\mathcal{T}_S^{S(r-\frac{1}{2}\sigma^2)} \mathcal{B}_{S'}^{\sigma^2} \mathcal{T}_a^{Se^{S'}}\right)^5 f(a)\\
&= \underset{S'=0}{\text{大}}\ e^{-5r} \left(\mathcal{T}_{S'}^{r-\frac{1}{2}\sigma^2} \mathcal{B}_{S'}^{\sigma^2} \mathcal{T}_a^{Se^{S'}}\right)^5 f(a)
\end{aligned}$$

After this transformation the initial stock price S can be considered as a constant. Any change is reflected by S' on logarithmic scale. The S' is initialized to zero with $\text{大}_{S'=0}$.

4.3 Numerical solution

When it comes to numerical solution, very often a huge gap between a mathematical concept with compact results and a computer implementation that is confronted with the limitations of computability opens. The incompatibility between math and programming is a consequence of very different concepts that can not be represented in each others world.
Chapter 1 on multiscale calculus offers cure for at least one source of such problems. A computational data type for mathematical functions and operators was introduced. Based on this implementation a wide range of common operators was made available to a computer program. In particular we have seen code for all the operators that are involved in our financial model. Hence, the transformation from operator notation to computer code should require hardly more than a transcription of each operator into the corresponding implementation.

4.3.1 Integration

Numeric integration is the standard method for the determination of an Asian option's price. Hence, the intgeration method will be the first that is turned into computer code. The integral representation, as developed in section 4.2.2, is a five dimensional integral over an integrand function to which several $\mathcal{T}$ operators are applied. We can write the complete integrand A as a sequence of initial values, the transformations $\mathcal{T}$ and final evaluation function.

$$A = \underset{\substack{S=1.0\\a=0.0}}{大} \left(\bigodot_{i=1}^{5} \mathcal{T}_S^{S\left(-\frac{2}{5}G^{-1}\left(\mathrm{x}_i+\frac{\Delta \mathrm{x}_i}{2}\right)-\frac{3}{50}\right)} \mathcal{T}_a^{S} \right) e^{-1/10} \max\left(\frac{a}{5}-1,0\right) \tag{4.10}$$

The argument to the inverse Gaussian distribution G^{-1} is shifted by half the resolution, such that we can later apply an integration method that starts at zero. Since we have already discussed the transcription of all involved operators the computer code is straight forward. It is assumed that all occurring numbers are automatically casted to the constant function (see 1.2).

```
S = 1.0
a = 0.0
FOR dim = 1 TO 5
  S = S*Exp(-2/5 * Invnorm(x(dim) + Delta(dim, x(dim))/2) -3/50)
  a = a+S
END
A = Exp(-1/10) * Max(a/5 - 1.0, 0)
```

Now, we apply the integral operators to the integrand. For the univariate integration the standard rule $\int \cdot \Delta \mathrm{x}$ was developed in section 1.3.2. This might not be the best integration rule available, but the integrand function is not smooth and many optimized quadrature rules will not necessarily work properly.

$$I = \int_0^1 \cdots \int_0^1 A \, \Delta \mathrm{x}_1 \cdots \Delta \mathrm{x}_5 \tag{4.11}$$

The integral has five dimensions and it is advisable to apply the sparse grid operator $\boxplus$ from section 1.5.4 to perform an efficient evaluation on heigh dimensions.

$$\mathsf{Z}_1^n \cdots \mathsf{Z}_5^n \underset{\{1,\cdots,5\}}{\boxplus} I \tag{4.12}$$

The accuracy level is determined by application of dilatation operators. The result does converge with with increased zoom level, but results could certainly be better.

Term n	3	4	5	Analytic
$(\Theta\mathcal{T}_a^S)^5\, a$	4.60	4.69	4.74	4.80
$(\Theta\mathcal{T}_a^S)^5\, a^2$	29.7	32.1	33.73	37.2
$(\Theta\mathcal{T}_a^S)^5\, a^3$	120	174	224	955
$(\Theta\mathcal{T}_a^S)^5 \max(a/5-1,0)$	7.0	29.3	23.2	–

Obviously the result for a^3 causes serious problems to the computational method. The problem is in part caused by rounding errors in the floating point arithmetics. Even the Monte-Carlo method does not converge to the correct result, unless floating point numbers with significantly more than 64 bits are used.

4.3.2 Standard scale

After the most common implementation approach was discussed we turn to the most straight forward approach. Given that you have set up the operator term for your financial product and given an implementation for the common operators, nothing is easier than just applying all the operators one by one.

$$\underset{\substack{S=1.0\\a=0.0}}{\text{大}}\left(e^{-r}\mathcal{T}_S^{rS}\mathcal{B}_S^{(\sigma S)^2}\mathcal{T}_a^S\right)^5 \max\left(\frac{a}{5}-1,0\right) \tag{4.13}$$

The continuous operators $\mathcal{T}$ and $\mathcal{B}$ do solve complex partial differential equations. In a discrete setting it is sufficient approximate them by T and B as defined in sections 1.4.3 and 1.4.2.

$$\approx \underset{\substack{S=1.0\\a=0.0}}{\text{大}}\left(e^{-r}T_S^{rS}\mathcal{B}_S^{(\sigma S)^2}T_a^S\right)^5 \max\left(\frac{a}{5}-1,0\right) \tag{4.14}$$

Before we can feed this into our multiscale algorithm we must get rid of the non zero initial value for S and determine the resolution with the application of the dilatation operator Z.

$$\cong \text{大}\,\mathsf{Z}^n T_S^1\left(e^{-r}T_S^{rS}\mathcal{B}_S^{(\sigma S)^2}T_a^S\right)^5 \max\left(\frac{a}{5}-1,0\right) \tag{4.15}$$

The convergence of this implementation is much better. The number of dimensions is lower, with only a and S. The polynomial order is better as well. Linear functions can be represented precisely with the linear interpolation method even on the coarsest scale.

Term n	0	1	2	Analytic
$(\Theta\mathcal{T}_a^S)^5\, a$	4.80	4.80	4.80	4.80
$(\Theta\mathcal{T}_a^S)^5\, a^2$	36.4	37.1	37.2	37.2
$(\Theta\mathcal{T}_a^S)^5\, a^3$	362	415	434	955
$(\Theta\mathcal{T}_a^S)^5 \max(a/5-1.0,0)$	23.0	25.0	24.76	–

The result behaves like function and it is just easy to evaluate the problem for different intial parameters. By changing the power of the repeated operator sequence, we can trace the intermediate computational steps. The results are visualized in figure 25.

4.3.3 Logarithmic scale

On logarithmic scale the transcription of operators is essentially the same as on standard scale. We even don't need to approximate our operators $\mathcal{T}$ and $\mathcal{B}$, since they have real valued exponents and can directly be translated into congruent discrete operators. Please keep in mind, that no further configuration

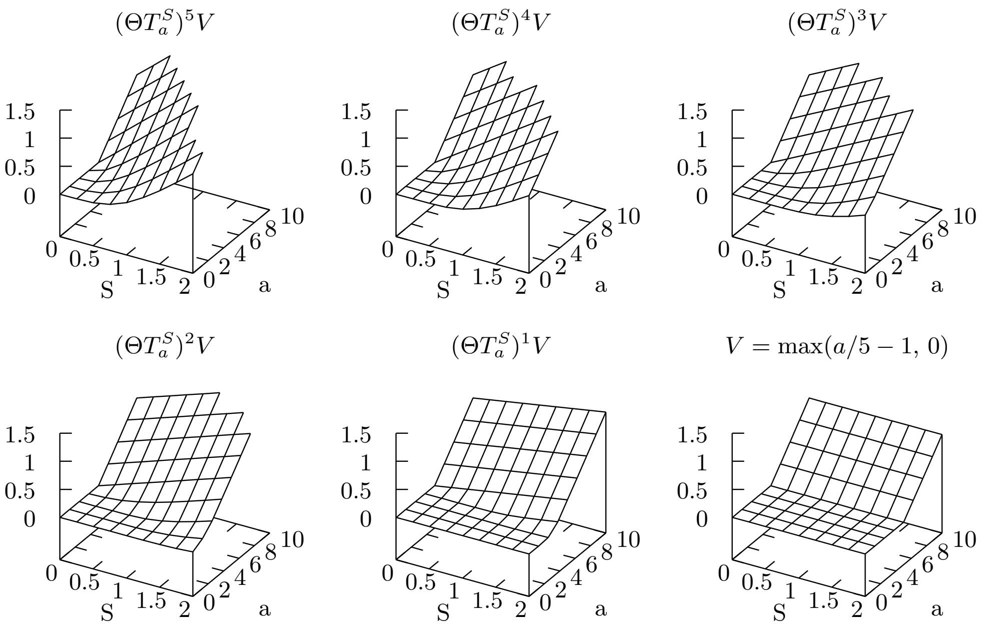


Figure 25: Value profile of the Asian option. The evaluation is performed backwards in time. The last picture in this sequence shows the final payment V and to which the operators are successively applied. The first picture shows Asian option price that depends on the initial value for the counter a and current spot price S.

was necessary to implement this term as a multiscale function. Domain borders are adjusted automatically and the resolution can always be determined with Z operators. The following term is just repeated from (4.9) and evaluated with the numerical scheme 大.

$$\text{大}\Pi(\Theta)c \cong \text{大}\mathsf{Z}^n e^{-5r} \left(T_{S'}^{r-\frac{1}{2}\sigma^2} B_{S'}^{\sigma^2} T_a^{1e^{S'}} \right)^5 \max\left(\frac{a}{5} - 1, 0\right) \tag{4.16}$$

The polynomial accuracy is lower on logarithmic scale and linear functions can no longer be captured precisely. A hard case is again the function a^3, with an error of more than 50% due to unprecise floating point representation. Fortunately there are no options with such a high curvature in the pay off. The resulting option price seems to settle around 24.7.

Term n	0	1	2	Analytic
$(\Theta \mathcal{T}_a^S)^5\, a$	4.85	4.80	4.80	4.80
$(\Theta \mathcal{T}_a^S)^5\, a^2$	42.6	36.8	37.0	37.2
$(\Theta \mathcal{T}_a^S)^5\, a^3$	722	412	425	955
$(\Theta \mathcal{T}_a^S)^5 \max(a/5 - 1, 0)$	25.5	24.8	24.71	–

4.4 Further tuning

The programming interface to multiscale functions works well in encapsulating some of the difficulties of numeric programming. Complex operator terms could be evaluated on the domain of mathematical functions without bothering about the internals of the functional implementation. However, it is sometimes unavoidable to trace down the computational procedure. Speed bottlenecks or sources of errors can only be detected after deeper research. We will now analyse and tune the code purely through our proxy operators, without having to dig in elementary program code.

4.4.1 Evaluation nodes

In order to tackle any speed problem connected to the evaluation of a multiscale function some measurements and visualizations can help detecting its source. Concerning speed it is highly relevant, how often and where a function is evaluated. If we have an operator term of the form $\Theta^5 f$ it results in a function and we have tight control over where we want to evaluate it. However, one call to $\Theta^5 f$ will be followed by more than one call to $\Theta^4 f$ at possibly very different coordinates. Each call to $\Theta^4 f$ will in turn generate calls to $\Theta^3 f$. Eventually a single call to the whole operator term $\Theta^5 f$ causes an abundance of calls to the plain function f. It is very important to understand this calling sequence.

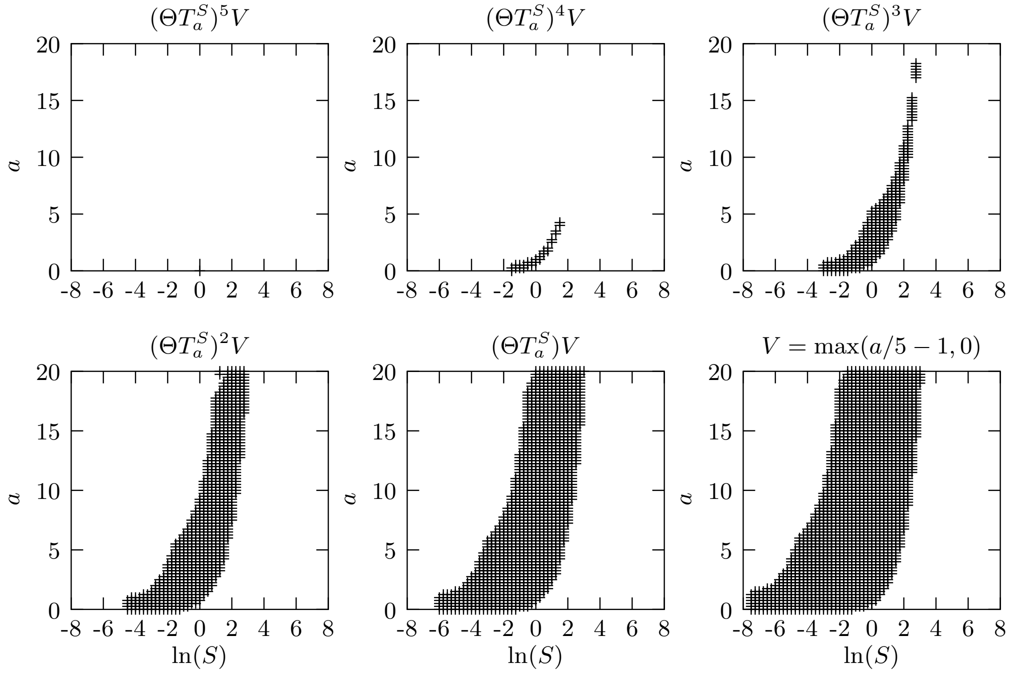


Figure 26: Calling sequence for the Asian option pricer. An evaluation of the operator term $(\Theta T_a^S)^5 V$ at one point, marked by a single cross in the first plot, causes several calls to subterm $(\theta T_a^S)^4 V$, marked in the second plot, and so on. Finally, the whole operator term requires multiple evalutations of the original function V, plotted finally.

Figure 26 visualizes the calling sequence for the Asian option pricer. This kind of graphics can be printed by the tracing proxy as discussed in 1.6.3. The initial term is only called once at the coordinates $(0, 0)$. Although not plotted to the full extent, the values on the a-axis do eventually reach very high values. While the S'-axis is scaled logarithmically, the a-axis is not. When the stock price $\ln(S)$ goes to 8, a is increased by $e^8 = 2981$. A large part of the function samples lie far out.

4.4.2 Local adaptivity

A more careful placement of evaluation nodes can be achieved by local adaptivity. In section 1.5.3 an a posteriori adaptive algorithm was introduced, which can be applied to a function just in the same way as any operator. Adding this kind of adaptivity to an existing algorithm has never been easier.
The first thing that we have to do is to set up an error estimation function. Based on the result of the refined solution Z and an interpolated guess W an error estimation E must be derived. A mixture between an absolute and a relative error is appropriate in this case.

$$E = \frac{|\mathsf{Z} - W|}{|\mathsf{Z}| + 1} \tag{4.17}$$

Configured with this error estimator we can apply the adaptive scheme A in both directions. The adaptivity can be placed at every position where it seems appropriate. In this example, five adaptive layers are inserted.

$$\left(\Theta \mathcal{T}_a^S A_1 A_2\right)^5 \max\left(\frac{a}{5} - 1, 0\right) \tag{4.18}$$

Figure 27 shows the effect of the adaptivity on the evaluated nodes. The parameters are $\epsilon = 0.005$ and $\alpha = 0$. Clearly the computational effort is focused on the singularity of the function. Two effects are worth mentioning. Firstly, the adaptive scheme never gets courser than the initial uni-spaced grid. Secondly, some extra effort for the adaptive scheme is required, since the space is more coarsely sampled before evaluation of a precise position. The difference becomes very clear in the second plot. Adaptivity should better have been switched off here. Also the last picture, where adaptivity interpolates a lot of values, but surely consumes more time for the interpolation than the simple maximum function would require.

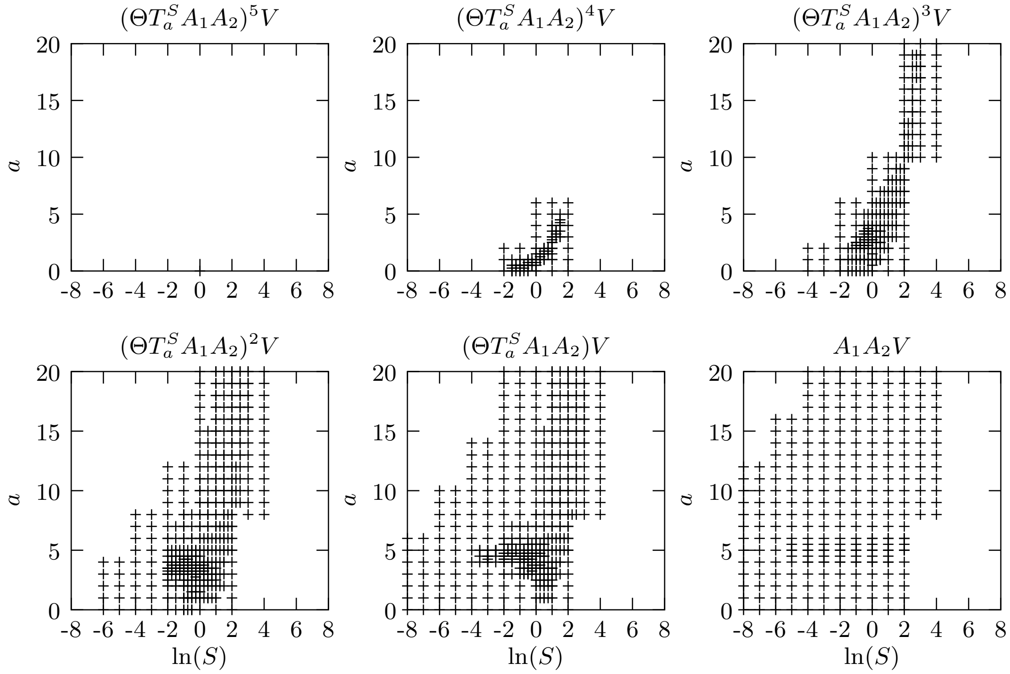


Figure 27: Evaluation nodes of the adaptive scheme. Compared to the previous figure evaluated points are sampled more coarsly where the function is smooth.

5 Conclusion

This document presented a case for the use of operators in finance and numerics. Strictly speaking, operators are functions that map functions onto functions. However, this thesis presented many applications of operators which gave this mathematical concept much more practical meanings. First, the well known operators integration and differentiation were redefined to work on a computer implementable data structure as a domain, on which further operations, such as adaptivity and sparse grid evaluation could be seen as operators too. A unified concept for the definition of the mathematical model and the numerical procedure is certainly desirable, since the two can hardly be viewed separately from each other. However, few methods essential for effective scientific computing were not explicitly presented but are implementable by principle. These include implicit PDE and iterative methods and multi-grid. A much more complete coverage of required operators was reached in the field of finance for the expression of financial contracts and products. Based on operators for elementary events a chronological sequence of operators could express all commonly traded contracts, which was previously not possible through neither technical nor mathematical means. Finally, a method was given to express stochastic processes in a way can be directly reduced to mathematical operations on the domain of functions, although it meant that some models could only be expressed as approximations. By and large this work demonstrated the applicability of operators for financial modeling as well as their applicability for the controlling of computational routines. Maybe the work with functions as data structures may once become as common place as matrices and vectors are used in numerics today.

Bibliography

[Bac00] L. Bachelier. Théorie de la spéculation. *Gauthier-Villars*, page 70 pp, 1900.

[BD03] H. J. Bungartz and S. Dirnstorfer. Multivariate quadrature on adaptive sparse grids. *Computing*, 2003.

[Bey91] G. Beylkin. Multiresolution analysis and fast numerical algorithms. *A draft of INRIA lecture notes*, 1991.

[Bey92] G. Beylkin. On the representation of operators in bases of compactly supported wavelets. *SIAM Journal on Numerical Analysis*, 1992.

[Bol86] T. Bollerslev. *Generalised Autoregressive Conditional Heteroskedasticity.* Journal of Econometrics 31, 1986. 307-27.

[BS73] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–659, 1973.

[Bun98] H. J. Bungartz. *Finite Elements of Higher Order on Sparse Grids.* Shaker Verlag, 1998.

[CR76] J. C. Cox and S. A. Ross. The valuation of options for alternative stochastic processes. *Journal of Financial Economics*, 3:145–166, 1976.

[Dir04] S. Dirnstorfer. On the representation of trading strategies and financial portfolios. In *intelligent finance*, ISBN 187685118X, pages 131–143, 2004.

[fpm04] Fpml 4.0 specification. Technical report, www.fpml.com, 2004.

[GMPZ04] F. Günther, M. Mehl, M. Pögl, and C. Zenger. A cache-aware algorithm for pdes on hierarchical data structures based on space filling curves. *SIAM Journal on Scientific Computing*, 2004.

[HJM92] Heath, Jarrow, and Morton. Bond pricing and the term structure of interest rates: A new methodology for contigent claims valuation. *Econometrica*, 1992.

[Hua98] K. Huang. *Quantum Field Theory: From Operators to Path Integrals.* John Wiley & Sons, Inc., 1998.

[Hul02] J. C. Hull. *Options, Futures, and Other Derivatives.* Pearson US Imports & PHIPEs, 2002.

[IK98] K. Inui and M. Kijima. A markovian framework in multi-factor heath-jarrow-morton models. *Journal of Financial and Quantitative Analysis*, 1998.

[JES00] Simon Peyton Jones, Jean-Marc Eber, and Julian Seward. Composing contracts: an adventure in financial engineering, 2000.

[JGT01] M.Griebel J. Garcke and M. Thess. Data mining with sparse grids. *Computing*, 67, 2001.

[JS93] B. Jawerth and W. Sweldens. An overview of wavelet based multiresolution analyses, 1993.

[KO01] B. K. Ksendal and Bernt Oksendal. *Stochastic Differential Equations: An Introduction With Applications*. Springer-Verlag, 2001.

[Mun98] C. Munk. The heath-jarrow-morton models of the term structure of interest rates. *Lecture notes*, December 1998.

[QMRLUF03] J. J. Quesada-Molina, J. A. Rodríguez-Lallena, and M. Úbeda-Flores. What are copulas? *Monografías del Semin. Matem. García de Galdeano*, 27:499–506, 2003.

[Ros99] S. M. Ross. *An Introduction to Mathematical Finance*. Cambridge university press, 1999.

[Sam65] P. A. Samuelson. Rational theory of warrant pricing. *Industrial Management Review*, 6:13–31, 1965.

[Shr97] S. E. Shreve. Stochastic calculus and finance, 1997.

[TR00] D. Tavella and C. Randall. *Pricing financial Instruments The Finite Difference Method*. John Wiley & Sons, Inc., 2000.

[Zag02] R. Zagst. *Interest Rate Management*. Springer Finance, 2002.