An object oriented approach to idempotent analysis: Integral Equations as Optimal Control Problems

Paola Loreti and Marco Pedicini

Abstract. Within the framework of generic programming, we implement an abstract algorithm for solution of an integral equation of second kind with the resolvent kernels method. Then, as an application of the idempotent analysis analog of resolvent kernels developed in [**LP01**], we apply the algorithm to the numerical solution of an optimal control problem with stopping time.

1991 *Mathematics Subject Classification.* Primary 68N19, 49L20, 46N10; Secondary 90C39, 45D05.

Key words and phrases. Idempotent Analysis, Object Oriented Programming, Optimal Control Theory.

1. Introduction

It is well known that discrete Bellman equations can be handled as linear over appropriate semirings. In a recent paper [**LP01**], the authors show how a class of Hamilton-Jacobi-Bellman equations arising from an optimal control problem with a stopping time can be treated as integral equations of Volterra type and usual methods of functional analysis can be applied in the approximation using resolvent kernels. After a discretization of the optimal control problem, using an appropriate algebraic structure for linearizing, the problem is rewritten as an integral equation and an analog of the Dirichlet formula for the idempotent calculus is found. The proofs of the main results in [**LP01**] are based on an integral representation of the solution, and here we extract computational and algorithmic contents of these results.

To explore the new method, we investigate the implementation within the framework of *generic programming*, where the implementation of a given algorithm can be applied to many data structures by using the STL (Standard Template Library) technology coming from Object Oriented programming. Templates are a widely-used C++ feature that in spite of simple behaviour may yield important implications for a developer. What a template permits is to mark a part of the code (including classes, methods, functions) as being a template for those classes, methods or functions that have sockets in them, where later on we can place constants like names for data structures or numbers. In practice, we may write type generic code realizing the quote "write once, run anywhere", in our specific case we write templates generic with respect to the arithmetical structure and after a discretization, specialize on floating point numbers. We propose the implementation of a library of templates (Kernel Template Library or KTL for short) realizing generic integration and computation of resolvent kernels on (semi)rings. Then, by specifying a particular semiring, we automatically obtain a program solving integral equations over a given semiring. By choosing Min-Plus algebra and using the analog of resolvent kernels in [**LP01**], we apply this method to compute the numerical approximation of the optimal control problem. On the other hand, we may apply the same template library to solve integral equations, simply by changing the definition of a semiring. The application of idempotent analysis to generic programming is discussed in [**LM00**], see also references therein.

In section 2, we recall some known results on Volterra equations [**VP36**], on idempotent analysis [**Mas87, LM98, KM97**], and on the analog of resolvent kernels approach for the idempotent approximation of Hamilton-Jacobi-Bellman equations [**LP01**].

In section 3, we introduce an implementation of Volterra resolvent kernels algorithm for integral equations. This implementation consist of a pair of templates for defining a scalar product, and a derived notion of definite integration. For the sake of simplicity we don't implement our prototype in C++ but in Mathematica to ease the mathematical understanding of the implementation.

Then, this integration is used to define iterated kernels, as in the usual Volterra approximation. All these definitions are generic and based on abstract arithmetical operations and topological structure that are later introduced in order to discretize the set.

In fact in section 4, we apply the two templates in two distinct ways by considering two different arithmetical structures on real numbers:

(1) the first case consists in applying the algorithm to a particular Hamilton-Jacobi-Bellman equation. We implement the arithmetical structure of real numbers and then the topology required by the discretization of the integral operation. As pointed out by many authors, the required structure is a Min-Plus algebra:
$$\mathbb{R}_{\min,+} = (\mathbb{R} \cup \{+\infty\}, \min, +, +\infty, 0);$$
(2) the second structure we consider is the usual ring of real numbers
$$\mathbb{R} = (\mathbb{R}, +, \cdot, 0, 1)$$
with a scalar product obtained by a discretization of the domain interval in intervals with constant diameter h. In this case, templates give the usual resolvent kernel approximation of the solution of an integral equation.

The main difficulty is to give an abstract algorithm which performs the integration, depending on the data structure and its discretization. This is obtained by using a very simple quadrature rule based on Riemann sums, in section 3.1.

In section 5, we show how to apply the templates and the semiring previously introduced, to run on several numerical examples.

2. Background

2.1. Iterated Kernels. Let us recall known facts on Volterra equations. We consider the Volterra equation of the second kind:
$$u(x) = \psi(x) + \lambda \int_0^x u(y)\, K(y,x)dy,$$
where the function K is called the *kernel* of the equation. It is possible to find a solution of this equation as a power series in λ. In fact, let us consider the iterated kernels:
$$K^{(n)}(y,x) = \int_y^x K^{(1)}(y,z)K^{(n-1)}(z,x)dz \qquad n \geq 2$$
and $K^{(1)}(y,x) = K(y,x)$.

Let us define an iterative sequence $\{u_n(x)\}_{n\in\mathbb{N}}$ by the following
$$u_n(x) = \begin{cases} \psi(x) & n = 0, \\ \psi(x) + \lambda \int_0^x u_{n-1}(y)\, K(y,x)dy & n > 0. \end{cases}$$
From this we get
$$\begin{aligned} u_n(x) &= \psi(x) + \sum_{s=1}^{n} \lambda^s \int_0^x \psi(y)\, K^{(s)}(y,x)dy = \\ &= u_{n-1}(x) + \lambda^n \int_0^x \psi(y)\, K^{(n)}(y,x)dy. \end{aligned}$$
Thus, passing to the limit as $n \to \infty$ we have the following power series in λ
$$\begin{aligned} u(x) &= \psi(x) + \sum_{s=1}^{\infty} \lambda^s \int_0^x \psi(y)\, K^{(s)}(y,x)dy \\ &= \psi(x) + \lambda \int_0^x \psi(y) R(y,x;\lambda)dy, \end{aligned}$$

where the, so called, *resolvent kernel* is given by

$$R(y,x;\lambda)=\sum_{s=1}^{\infty}\lambda^{s-1}K^{(s)}(y,x).$$

2.2. Few facts on idempotent analysis. The classical example of semiring is $\mathbb{R}_{\min,+}:=(\mathbb{R}\cup\{+\infty\},\min,+)$, we denote by $a\oplus b:=\min(a,b)$ the minimum, and by $a\odot b:=a+b$ usual sum.

- idempotency $a\oplus a=a$;
- zero $\mathbf{0}:=+\infty$;
- identity $\mathbf{1}:=0$;
- idempotency implies, in a standard way, a linear order

$$a\preceq b \text{ if and only if } a\oplus b=b$$

- moreover, a metric can be provided $\rho(a,b):=|e^{-a}-e^{-b}|$.
  In this case, we say $(\mathbb{R}_{\min},\rho)$ a metric semiring.
- semimodules of functions: $X\to\mathbb{R}_{\min,+}$,
- scalar product:

$$\langle f,g\rangle:=\inf_{x\in X}(f(x)+g(x))$$

- linear operators:

$$(Af)(x)=\inf_{y\in X}(K(x,y)+f(y))=\int_X^{\oplus}K(x,y)\odot f(y)dy$$

- idempotent measures:

$$\mu:X\to\mathbb{R}_{\min,+} \text{ where } \mu(A)=\inf_{a\in A}\mu(a)=\langle\delta_A,\mu\rangle$$

  where δ_A is the characteristic function of the set A defined as

$$\delta_A(x)=\begin{cases}\mathbf{1} & \text{if } x\in A\\ \mathbf{0} & \text{if } x\notin A\end{cases}$$

- idempotent integral, for a given idempotent measure μ on X:

$$\int_X^{\oplus}f(x)dx:=\langle f,\mu\rangle.$$

- linearity

$$\int_X^{\oplus}f(x)\oplus g(x)dx=\int_X^{\oplus}f(x)dx\oplus\int_X^{\oplus}g(x)dx$$

- additivity

$$\int_{X_1\cup X_2}^{\oplus}f(x)dx=\int_{X_1}^{\oplus}f(x)dx\oplus\int_{X_2}^{\oplus}f(x)dx$$

- Fubini Theorem:

$$\int_X^{\oplus}\int_Y^{\oplus}\phi(x,y)dydx=\int_Y^{\oplus}\int_X^{\oplus}\phi(x,y)dxdy$$

- Dirichlet Formula:

$$\int_X^{\oplus}\int_{F(x)}^{\oplus}\phi(x,y)dydx=\int_{F(X)}^{\oplus}\int_{F^{-1}(y)}^{\oplus}\phi(x,y)dxdy.$$

2.3. Idempotent approximation of Hamilton-Jacobi-Bellman equations. V. P. Maslov observed the analogy between viscosity solutions and linearity of Bellman equation in appropriate idempotent semirings [**Mas87, MS92**].

We consider the Hamilton-Jacobi-Bellman equation associated to a particular optimal control problem:

$$\max(u(x) - \psi(x), \lambda u(x) + \max_{a\in A}\{-b(x,a)\cdot Du(x) - f(x,a)\}) = 0 \tag{2.1}$$

and its discretized

$$u_h(x) = \min\left(\psi(x), \inf_{a\in A}\left((1-\lambda h)u_h(x+hb(x,a)) + hf(x,a)\right)\right), \tag{2.2}$$

under the assumptions considered in [**LP01**].

Note that the generalized stationary finite dimensional Bellman equation can be written as

$$S = HS \oplus F$$

where $\oplus$ gives the structure of a linear equation in the semiring

$$\mathbb{R}_{\min,+} := (\mathbb{R}\cup\{+\infty\}, \min, +).$$

Equation (2.2) results to be an *integral equation* if regarded in $\mathbb{R}_{\min,+}$ with the metric and the integral defined as in section 2.2,

$$u_h(x) = \psi(x) \oplus \tilde{\lambda}\int_{B(x,A)}^{\oplus} u_h(\xi)\odot G_h(x,\xi)d\xi$$

where $\tilde{\lambda} = 1-\lambda h$, $B(x,a) = x + hb(x,a)$,

$$G_h(x,\xi) = \frac{h}{1-\lambda h}\min\{f(x,a) | a\in A, \xi = x + hb(x,a)\}. \tag{2.3}$$

Resolvent kernels appeared as a natural approximation for non homogeneus integral equations, as it is the case for our equation:

$$u_h(x) = \psi(x) \oplus \tilde{\lambda}\int_{B(x,A)}^{\oplus} u_h(\xi)\odot G_h(x,\xi)d\xi.$$

So, we obtain

$$u_h(x) = \psi(x) \oplus \bigoplus_{n=1}^{\infty}\tilde{\lambda}^n\int_{B^n(x,A)}^{\oplus}\psi(t)\odot K^{(n)}(t,x)dt$$

where the *iterated kernels* are

$$K^{(1)}(t,x) = G_h(x,t) \qquad t\in B^{(p)}(x,A), p\geq 1$$
$$K^{(n+1)}(t,x) = \int_{B^{-1}(t)}^{\oplus} K^{(1)}(t,\xi)\odot\frac{K^{(n)}(\xi,x)}{\tilde{\lambda}}d\xi \quad t\in B^{(p)}(x,A), p\geq n+1$$

and $B^n(x,A) = \{B(\dots B(B(x,\alpha_1),\alpha_2),\dots,\alpha_n)\}_{\alpha_i\in A}$.

From the Volterra equation we find a p-step approximation of the solution which corresponds to the dynamic programming principle:

$$u_h(x) = \psi(x) \oplus \bigoplus_{n=1}^{p}\tilde{\lambda}^n\int_{B^n(x,A)}^{\oplus}\psi(t)\odot K^{(n)}(t,x)dt \oplus$$
$$\oplus\,\tilde{\lambda}^{p+1}\int_{B^{p+1}(x,A)}^{\oplus} u_h(t)\odot K^{(p+1)}(t,x)dt.$$

As a consequence of this principle, in [**LP01**] we proved that the idempotent series is convergent, and uniqueness of the value function.

3. Overview of the implementation of Resolvent Kernels Algorithm

In this section we present an implementation at a prototyping stage, nevertheless it illustrates ideas in our approach well. This prototype is implemented by using Mathematica, in order to take advantage of the mathematical features which are present in the language. Our final target is to obtain a standard template library in C++, we plan to derive it from MTL (Matrix Template Library) a template library for matrix calculus. As an instance we consider to use in the future MathCode, a compiler from Mathematica code to C++.

In our implementation, we provide two generic templates required for the implementation of integral kernels. These templates are automatically adapted when used with usual real numbers or within min-plus algebra.

Then, we provide two modules defining operations and constants on the real numbers structure, and functions used for the discretization of intervals. This part is introduced to realize the proper instatiation of the arithmetical structure.

The second part of the implementation consists just in the definition of the particular problem we aim to solve. So, we specify the integral equation of second type by the kernel and its constant term function (in the analog of the optimal control problem, they can be derived from dynamics, the cost function and the stopping time cost function). At the very end of the program, the approximate solution of the integral equation is computed (and in the analog of the optimal control problem, this corresponds to an approximation of the value function). Now, let us enter into details on the first part of the implementation (the second one is treated in section 4, where we give two examples of the module relatively to the definition of both the min-plus algebra and the real numbers with usual operations), and in section 5 we run several numerical tests.

Kernel Template Library (`KTL.ma`) contains the two templates building scalar product and resolvent kernels:

⟨*code/KTL.ma*⟩≡
  ⟨*Template for the scalar product*⟩
  ⟨*Template for resolvent kernels*⟩

3.1. Approximate Integration on Real Numbers. In order to give a generic algorithm for the approximation of the two versions of integral we consider in this paper, we need abstract properties that they share. A crucial notion is that we can build them starting from a function $\mu : 2^{\Omega} \to \mathbb{R}$ completely additive, that is

$$\mu(\bigcup_{i\in I} B_i) = \bigoplus_{i\in I} \mu(B_i)$$

for any family $\{B_i\}_{i\in I}$ of measurable subsets of Ω, and it is called an $\mathbb{R}$-measure on Ω. When $\oplus$ is the sum, this corresponds to a well known property defining Lebesgue measure. In the idempotent case within min-plus algebra, where $\oplus$ is min, we have that $\bigoplus_{i\in I} \mu(B_i) = \inf_{i\in I} \mu(B_i)$.

The idempotent integral shares many properties with the conventional integral, the most remarkably one is linearity; but, with respect to its approximation, we note that in both cases, it is the limit of Riemann's integral sums. Let f be continuous on the unit sphere $B \subset \mathbb{R}^n$ and let $\{B_i\}_{i\in I}$ be a partition into intervals of diameter $\leq k$, that is $B = \bigcup_{i\in I} B_i$ and $B_i \cap B_j = \emptyset$ for all $i \neq j$. Then

$$\int_B^{\oplus} f(c)d\mu = \lim_{k\to 0} \bigoplus_{i\in I} f(x_j) \odot \mu(B_j),$$

where x_j is an arbitrary point in B_j and $\odot$ denotes multiplication (resp. sum) in the classic (resp. idempotent) case, [**KM97**].

In order to implement our measure on a measurable space X we consider the characteristic function of any measurable set $E \subset X$

$$\delta_E(x) = \begin{cases} \mathbf{1} & \text{if } x \in E \\ \mathbf{0} & \text{if } x \notin E \end{cases}$$

whose implementation relies on the definition of a boolean function which tests if a given point x *belongs* to a given set X.

For every pair of functions f and g we define

$$\langle f, g\rangle := \bigoplus_{i=1}^{n} f(x_i) \odot g(x_i) \odot \mu(B_i), \tag{3.1}$$

from the scalar product and from the characteristic function we obtain

$$\int_B^{\oplus} f(x) = \langle f, \delta_B\rangle.$$

Let us explain functions used independently from the particular semiring, which provide the computation of Riemann's sums and the corresponding discretizazion of the sub-set $\Omega \subset \mathbb{R}$ based on a finite partition $\{B_i\}_{i\in I}$:

a) function $\delta_X(x)$ (`Delta[X_][x_]`) is defined starting from the function which tests $x \in X$ (`Belongs[x,X]`), this function is provided by the discretization module;
b) the measure is obtained by additivity on the measure of the element of the partition; this is defined in the discretitation module relatively to the considered semiring: in the idempotent case it is $\mu(B_i) = \mathbf{1}$ (`MuB[X_] := One;`), whilst in the classical real numbers structure we have $\mu(B_i) = diameter(B_i)$;
c) we need also a constant **1** function (`constOne[x_] := One`) where the definition of the identity is given in the semiring module;

d) we define the scalar product that is used to define the approximate integral as in equation (3.1)

$$\langle f, g\rangle = \bigoplus_{i\in I} f(x_i) \odot g(x_i) \odot \mu(B_i)$$

where I (`IntegralMesh`) is the indices family, x_i (`GetPoint[B[i]]`) is a random point in B_i, this function has to be defined in the discretization module;

e) from the scalar product we obtain also the measure of a generic (measurable) set X as the scalar product of the characteristic function of X and the constant to $\mathbf{1}$ function:

$\mu(X) = \langle \delta_X, \mathbf{1}\rangle$ (`Mu[X_] := ScalarProduct[ Delta[X], constOne];`)

f) in a similar way, the integral of the function f on a set A

$$\int_A^{\oplus} f(x)d\mu$$

is obtained as the scalar product of f with the characteristic function of A:

$\int_A^{\oplus} f(x)d\mu = \langle f, \delta_A\rangle$ (`Q[f_, A_] := ScalarProduct[f, Delta[A]]`)

⟨*Template for the scalar product*⟩≡

```
(* Template Class for Volterra Resolvent kernels method *)

(* Template for the scalar product definition *)

TemplateScalarProduct:=
(
 Delta[X_][x_] := Belongs[x, X];
 ScalarProduct[f_, g_] := Module[{x, i},
     CirclePlus @@ Map[
         CircleTimes @@ {f[x = GetPoint[B[#]]] , g[x],  MuB[#]} &,
         IntegralMesh
     ]
 ];
 constOne[x_] := One;
 Mu[X_] := ScalarProduct[ Delta[X], constOne];
 Q[f_, A_] := ScalarProduct[f, Delta[A]];
)
```

Let us summarize dependencies of this module: we need `Belongs`, `GetPoint`, `MuB`, and `IntegralMesh` from the discretization module of the semiring definition and `CirclePlus`, `CircleTimes`, `One` and `Zero` from the definition of the semiring.

3.2. Resolvent Kernels. This template is completely based on the scalar product definition (thus, on the discretization of the numeric domain and in particular on the computation of the discretized integral) and on the definition of the problem from which it uses the kernel $G(x, y)$, a function $\Gamma^{(n)}(x)$ which relates the point x to the integration domain at the n-th resolvent kernel approximation, implemented as `IntegralDomain[]`, and an analog function $\Delta^{(n)}(x, y)$ (implemented as `KernelDomain[]`) which relates the integration domain in the definition of the n-th iterated kernel to the points x and y.

Moreover, we define two private methods:

(1) the first function $h^{(k)}$ is the function to be integrated at the iteration k:

$$h^{(k)}(z, \langle x, y\rangle) = K^1(y, z) \odot \frac{K^{(k-1)}(z, x)}{\tilde{\lambda}}$$

```
H[z_, k_, A_] :=
  CircleTimes @@
    {
     K[1, A[[1, z]] ],
     lambdatilde^(-1) K[k - 1, { z , A[[2]] } ]
    };
```

(2) the second function `Kiterator`(n, K, ψ, x) compute the n-th approximation of the kernel with ψ as constant term at the point x:

$$\texttt{Kiterator}(n, K, \psi, x) = \tilde{\lambda}^n \int_{\Gamma^{(n)}(x)}^{\oplus} \psi(\xi) \odot K^{(n)}(\xi, x) d\xi$$

```
Kiterator[n_, K_, Psi_, x_] :=
   lambdatilde^n Q[
        (CircleTimes @@ { Psi[#]} , K[n, {# , x }]) &,
        IntegralDomain[n,x]
        ]
```

⟨*Private Methods*⟩≡

```
H[z_, k_, args_] := H[z, k, args] =
         CircleTimes @@ {
            K[1,{args[[1]],z}],
            lambdatilde^(-1) K[k - 1, {z,args[[2]]}]
         };

Kiterator[n_, K_, Psi_, x_] :=
         lambdatilde^n Q[
             (CircleTimes @@ {K[n, {# , x }], Psi[#]}) &,
             IntegralDomain[n,x]
         ];
```

Then iterated kernels are obtained by a recursive definition as:

$$\begin{cases} K^{(1)}(\langle y, x\rangle) = G(x, y) \\ K^{(k)}(\langle y, x\rangle) = \displaystyle\int_{\Delta^k(y,x)}^{\oplus} h^k(z, \langle y, x\rangle) dz. \end{cases}$$

⟨*Iterated Kernels*⟩≡

```
K[1, args_]  := K[1, args]
              = G[args[[2]], args[[1]]];

K[k_, args_] := K[k, args]
              = Q[H[#, k, args] &, KernelDomain[k-1, args ]];
```

The solution of the integral equation is computed with the help of `Kiterator` and it is inductively defined as

$$\begin{cases} u_h^{(0)}(x) = \psi(x) \\ u_h^{(n+1)}(x) = u_h^{(n)}(x) \oplus \tilde{\lambda}^n \displaystyle\int_{\Gamma^{(n)}(x)}^{\oplus} \psi(\xi) \odot K^{(n)}(\xi, x) d\xi \end{cases}$$

⟨*Approximate solution of the integral equation*⟩≡

```
u[0, x_] := Psi[x];

u[n_, x_] := u[n, x]
           = CirclePlus @@ {
                u[n-1 , x],
                Kiterator[n, K, Psi, x]
             };
```

Let us summarize dependencies of this template (for all functions hereby defined):

(1) on the real numbers data structure, specified in the semiring module, from which the $\oplus$ and $\odot$ are taken;
(2) on the kernel $G(x, y)$ and the integration domain function specified in the problem statement;
(3) on the integral approximation specified in the scalar product template.

⟨*Template for resolvent kernels*⟩≡

```
(* Template for the iterated Kernel definition *)
TemplateResolventKernel:=
(
 Print["initialiazing Resolvent Kernels Template"];
 ⟨Private Methods⟩
 ⟨Iterated Kernels⟩
 ⟨Approximate solution of the integral equation⟩
)
```

4. Semiring definitions and discretizations

The next two modules alternatively determine the particular algebraic structure we aim to use.

⟨*code/MinPlus.ma*⟩≡
⟨*Min-Plus Semiring*⟩
⟨*Min-Plus Discretization*⟩

⟨*code/Klassic.ma*⟩≡
⟨*Classic Semiring*⟩
⟨*Classic Discretization*⟩

4.1. Min-Plus Algebra. This structure is defined by two operations and two constants in the rest of the paper $\oplus$ denote the additive one, and $\odot$ the multiplicative one. Moreover, the two constants are denoted by $\mathbf{0}$ and $\mathbf{1}$.

First we consider the min-plus algebra $(\mathbb{R}, \min, +, +\infty, 0)$:

4.1.1. *Definition.*

⟨*Min-Plus Semiring*⟩≡

```
(* Definition of the basic structure      *)

(* algebraic structure and discretization *)
(* min-plus algebra                       *)
(* discrete measure space:  mu({x})=1     *)

Semiring[S_] := Module[{},
(
      CirclePlus := Min;
      CircleTimes := Plus;
      One := 0;
      Zero := Infinity;
      )]
```

4.1.2. *Discretization.* We consider the interval $\Omega = [a, b]$ and we consider a discretization built in the following way: let us fix k as the base diameter, we consider intervals $\Omega_i = [a_i, b_i]$ for $i = 0, \ldots, n-1$ where $a_i = a+ik$, $b_i = a+(i+1)k$ and $n = \lfloor (b-a)/k \rfloor$. Then for any a_i we consider the set of points reached using the dynamics of the optimal control problem in section 2.3:

$$\begin{cases} B^1[a_i, A] = \{x | x = a_i + hb(a_i, a), a \in A\} \\ \\ B^{n+1}[a_i, A] = \{x | x = \xi + hb(\xi, a), a \in A, \xi \in B^n(a_i, A)\} \end{cases}$$

we compute the discrete set of nodes $\tilde{\Omega}_s$ for any fixed integer s,

$$\tilde{\Omega}_s = \bigcup_{0 \leq i \leq n-1} \bigcup_{j=1}^{s} B^j[a_i, A]$$

from which we obtain a discrete measure space with all points reacheable from the initial points a_i:

$$\tilde{\Omega}_s = \{\omega_1 \leq \omega_2 \leq \cdots \leq \omega_m\}$$

where $m = |\tilde{\Omega}_s|$. Finally, we use sets $\{\omega_i\}$ to build by closure, the measure space. We assign measure $\mu(\{\omega_i\}) = \mathbf{1}$ for all $0 \leq i \leq m$ and we compute the idempotent integral by using the scalar product template. Note that in general, $\omega_1 \leq a$ and $b \leq \omega_m$. This construction is necessary to achieve the property: for every point $\omega_i \in \tilde{\Omega}_s$, $B[\omega_i, A] \subset \tilde{\Omega}_{s+1}$ in this way we know that points involved in the computation of the s-th iterated kernel at any point a_i belongs to $\tilde{\Omega}_s$.

⟨*Min-Plus Discretization*⟩≡

```
DiscretizationFunctions:=
(
  GetPoint[A_] := A[[1]];
  Belongs[x_, A_] := If[Count[A, x] >0 , One, Zero];
  B[i_] := {OmegaSet[[i]]};
  MuB[X_] := One;

(* definition of the state space *)
  B[1,x_]  := B[1,x]
           = Union[Flatten[Map[ (x + h b[x, #]) &, Ac]]] ;

  B[n_,x_] := B[n,x]
           = Union[Flatten[Map[ B[1,#] &, B[n-1,x] ]]] ;

  CreateSet[0] := CreateSet[0]
               = Union[Join[{0},Table[Omega[[1]]+ k j, {j,0,nk}]]];

  CreateSet[m_]:= CreateSet[m]
               = Union[Flatten[Table[
                   Table[B[i,{Omega[[1]]+ k j} ],{i,1,m}],{j,0,nk}]
                 ]] ;

(* generate reacheable points after the application of dynamics steps *)
  nk := nk
      = Floor[(Omega[[2]] - Omega[[1]]) / k ] ;
```

```
    steps = 20 ;
    OmegaSet = CreateSet[steps];
    n := n = Length[OmegaSet];
    IntegralMesh = Table[i, {i, 1, n-1}];
  )
```

4.2. Usual ring of real numbers.

4.2.1. *Definition.* This is the usual real numbers ring $(\mathbb{R}, +, \cdot, 0, 1)$:

⟨*Classic Semiring*⟩≡

```
  (* Definition of the basic structure      *)

  (* algebraic structure and discretization *)
  (* real number                            *)
  (* measure space: mu([a,b])=b-a           *)

  Semiring[S_] := Module[{},
   (
     CirclePlus := Plus;
     CircleTimes := Times;
     One := 1;
     Zero := 0;
   )
  ]
```

4.2.2. *Discretization.* Discretization in this case is very simple, we create a uniform partition of $\lfloor \frac{b-a}{k} \rfloor$ intervals $[a + i\,k, a + (i+1)\,k]$ of diameter k. Then, we define the function `GetPoint` as the function yielding always the left bound of the interval. As the function `Belongs` tests if a point belong to an interval we have that also the quadrature rule is valid only if $b \geq a$.

⟨*Classic Discretization*⟩≡

```
    GetPoint[A_] := A[[1]];
    Belongs[x_, A_] := If[(x >= A[[1]]) && (x<A[[2]]) , One, Zero];
    B[i_] := {Omega[[1]]+i k, Omega[[1]]+ k (i+1)};
    MuB[X_] := k;

    n=nk;
    IntegralMesh = Table[i,{i,0,nk-1}];
    OmegaSet := OmegaSet = Table[B[i], {i,0,nk-1}];
    OmegaDrawLight[j_,np_]:= OmegaSet ;
```

⟨*Instatiation of templates with real number structure*⟩≡

```
(* Template Instatiation *)
Clear[K]
Semiring[standard];
 Print["semiring initialized\n"];
DiscretizationFunctions;
 Print["discretization parameters initialized"];
TemplateScalarProduct;
TemplateResolventKernel;
```

5. Numerical Tests

We compare the behavior of numerical runs using the idempotent approach and a known case treated in Capuzzo-Dolcetta and Ishii [**CDI84**], and in Capuzzo-Dolcetta and Bardi book appendix by M. Falcone [**BCD97**].

Let us briefly recall some results on the approximation of Hamilton-Jacobi-Bellman equation.

We consider the approximate equation

$$u_h(x) = \min_a\{(1-\lambda h)u_h(x+hb(x,a)) + hf(x,a)\}. \tag{5.1}$$

From [**CDI84**], under suitable assumptions on the data, the value function u_h satisfies the following:

> *u_h converges uniformly to the unique viscosity solution u of the Bellman equation, as h goes to 0.*

Moreover, the following estimate was established

$$||u - u_h||_\infty \leq Ch^{1/2}$$

where C is a positive constant.

In [**Fal87**], a further step was done to approximate the unique viscosity solution by performing a discretization in space.

The following estimate holds true:

$$||u_h - u_h^{(k)}||_\infty \leq \frac{L_f}{\lambda - L_b}\frac{k}{h},$$

where we denote by u_h^k the approximate solution of the space-time discretized version of equation (5.1) and L_f and L_b are the Lipschitz constants of f and b respectively.

Finally, equation (5.1) can be solved by an iterative method

$$u_h^{(n)}(x) = \begin{cases} 0 & n = 0, \\ \min_a\{(1-\lambda h)u_h^{(n-1)}(x+hb(x,a)) + hf(x,a)\} & n > 0. \end{cases}$$

5.1. First case. Consider the dynamical system

$$\dot{X}(s) = -X(s) \cdot \alpha(s), \quad X(0) = x$$

with bounded controls:

$$|\alpha(s)| \leq 1.$$

We consider the value function

$$u(x) = \inf_{\alpha} \int_0^{\infty} -|X_x^{\alpha}(s)| e^{-2s}\, ds,$$

where $X_x^{\alpha}(t)$ denotes the state depending on the control α and on the initial position x.

The problem is to determine the value function $u(x)$ and the corresponding optimal control

- $u(x) = -|x|$ for every $x \in \mathbb{R}$,
- the optimal control is the constant function $\alpha = -1$.

In fact, for every admissible control α we have

$$-|X_x^{\alpha}(t)| = -|x| e^{-\int_0^t \alpha(s)\, ds} \geq -|x| e^t, \quad t \geq 0$$

hence

$$-\int_0^{\infty} |X_x^{\alpha}(t)| e^{-2t}\, dt \geq \int_0^{\infty} -|x| e^{-t}\, dt = -|x|.$$

with equality if and only if $\alpha(s) = -1$ for every s.

Let us consider the Hamilton-Jacobi-Bellman equation associated to the example

$$2u(x) + \max_{|a| \leq 1} \{a x u'(x) + |x|\},$$

equivalent to

$$2u(x) + |x| \cdot \big(|u'(x)| + 1\big) = 0 \quad \text{in} \quad \mathbb{R}.$$

From the equation we can find the optimal control just realizing

$$\max_{|a| \leq 1} \{a x u'(x) + |x|\}.$$

If $xu'(x) > 0$ we select $\alpha = 1$, while if $xu'(x) < 0$ we select $\alpha = -1$, so the optimal feedback is given by

$$\alpha(x) = \operatorname{sign}(xu'(x)).$$

We consider the approximate equation

$$u_h(x) = \min_a \{(1 - 2h) u_h(x - hxa) - h|x|\}.$$

The above equation can be solved by an iterative method

$$u_h^{(n)}(x) = \begin{cases} 0 & n = 0, \\ \min_a \{(1 - 2h) u_h^{(n-1)}(x - hxa) - h|x|\} & n > 0 \end{cases}$$

From this we get the approximate solution

$$u_h^{(0)} = 0, \qquad u_h^{(1)} = -h|x|,$$

$$u_h^{(2)}(x) = \min_a \{-(1 - 2h) h|x - hxa| - h|x|\}$$

Hence

$$u_2(x) = -2h|x| + h^2|x| + 2h^3|x|.$$

Note that the procedure can be iterated but numerical tests show that it slowly converges to the solution, and so a great number of iterations have to be performed in order to obtain a satisfactory approximation.

In figure 1, we compare the approximate value function obtained by the above described iterated scheme at successive four iterations with time step $h = 0.4$ and space step $k = 0.4$.

By the approximation formula for the value function we know:

$$u_h(x) = \inf_{\alpha_h^k} h \sum_{k=0}^{\infty} (1 - \lambda h)^k f(X_h^k, \alpha_h^k).$$

In this case, the infimum over the controls gives

$$f(X_h^k, \alpha_h^k) = -|x| e^{hk}$$

so that

$$u_h(x) = -|x| h \sum_{k=0}^{\infty} \left[(1 - 2h)e^h\right]^k = -|x| \frac{h}{1 - e^h + 2he^h}$$

The error between the value function and its approximate, for h small, is about $\dfrac{2h}{1+2h}$. By numerical test, we find similar approximation errors between the two methods (compare figures 1 and 2).

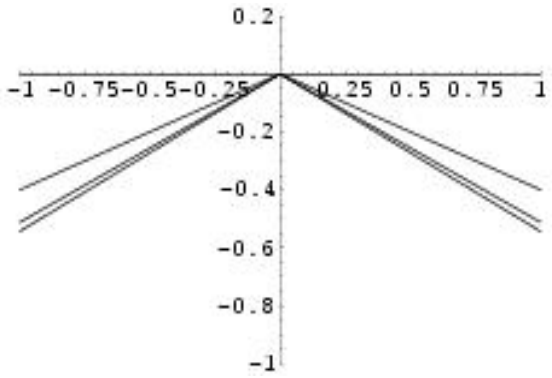

FIGURE 1. Numerical approximation of HJB, $h = 0.4$, $k = 0.4$

Now, we tackle the same problem by using templates above introduced for resolvent kernels. This requires to express the problem by means of an integral equation in an appropriate idempotent semiring $(\mathbb{R}_{\min,+})$.

We implement the optimal control problem by defining the problem statement part, i.e. the analog of the Hamilton-Jacobi-Bellman discretized equation is rewritten as an integral Volterra equation of second kind.

⟨*code/MinPlusExample.ma*⟩≡
  ⟨*The optimal control problem*⟩
  `<< KTL.ma`
  `<< MinPlus.ma`

⟨*Instatiation of templates with real number structure*⟩
⟨*Computation and visualization*⟩

By following section 2.1, equation (2.2) and [**LP01**], we assign the following parameters:

⟨*Optimal Control Problem definition*⟩≡

```
(* set of control values *)
   Ac = {-1, 1};

(* problem dynamics *)
   b[x_, a_] := -a x

(* cost function *)
   f[x_, a_] := -Abs[x]

(* stopping cost function *)
   Psi[x_] := 0

(* discount factor *)
   Lambda = 2
```

Then we fix the basic interval and we assign the discretization steps, so that functions $\Gamma^{(n)}$ and $\Delta^{(n)}$ can be derived from dynamics:

⟨*Discretization Parameters*⟩≡

```
(* defintion of the state space *)
  oa = -1; ob = 1;
  Omega = {oa, ob}

(* discretization parameters*)
  h = 0.4
  k = 0.4

(* integration domain funtions *)
  IntegralDomain[n_,x_] := B[n,x];
  KernelDomain[n_, args_]:= B[n,args[[2]]];
```

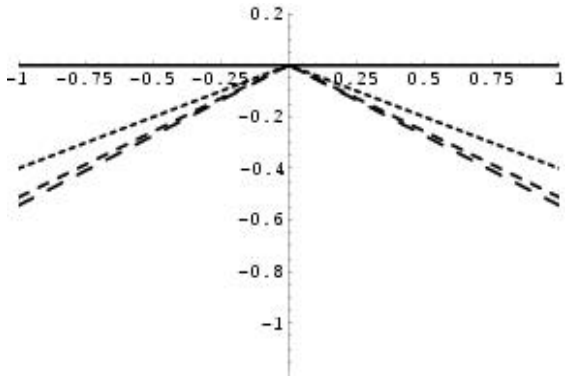

FIGURE 2. Graph of successive approximations (from 0 to 3) obtained with the KTL implementation, $h = 0.4$ and $k = 0.4$

From the problem dynamics and the cost functions we express the kernel for the Volterra integral function, see equation (2.3). More specifically, from function b and from the set of control values A, we obtain the kernel $G(x, y)$ of the integral equation and so, the approximate solution (the value function $u_h(x)$) can be computed by the resolvent kernels method. Here, we need also to define two auxiliary functions used in the definition of G.

⟨*Definition of the kernel analog*⟩≡

```
(* definition of the kernel *)
lambdatilde = 1 - Lambda h;

G[x_, y_] := Module[{XA},
        h/lambdatilde If[ ( (XA = SelectControls[x, y, Ac]) == {} ),
        (
         Zero
),
         f[x, ArgMin[f[x, #] &, XA]]
]]

SelectControls[x_, y_, A_] := Select[ A , ( x + h b[x, #] == y) &];

ArgMin[f_, A_] := Sort[Map[{f[#], #} &, A]][[1]][[2]]
```

Running the code with this example gives the picture of the successive approximations in figure 2 (with considered control values $\{-1, 1\}$)

⟨*The optimal control problem*⟩≡

```
(* Volterra Integral Equation aliasing an Optimal Control Problem *)
```

⟨*Definition of the kernel analog*⟩
⟨*Discretization Parameters*⟩
⟨*Optimal Control Problem definition*⟩

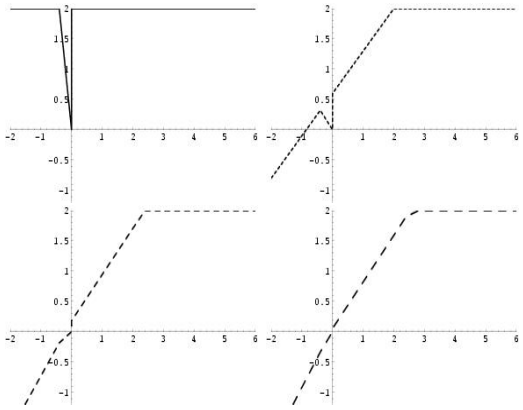

FIGURE 3. $u_h^{(n)}$ for $n = 0, 1, 2, 3$, and $h = 0.7$, $k = 0.4$,

5.2. Further Examples. Consider the dynamical system

$$\dot{X}(s) = -X(s) \cdot \alpha(s), \quad X(0) = x$$

with bounded controls:

$$0 \leq \alpha(s) \leq 1.$$

We consider the value function

$$u(x) = \inf_{\alpha,\theta} \int_0^\theta X_x^\alpha(s) e^{-s} \, ds + 2e^{-\theta},$$

where $X_x^\alpha(t)$ denotes the state depending on the control α and on the initial position x.

In the numerical test, we compute the solution of the integral equation with discretization parameters $h = 0.49$ and $k = 0.35$.

Exact solution of the control problem is

$$u(x) = \begin{cases} x & x \leq 0 \\ \dfrac{x}{2} & 0 < x \leq 4 \\ 2 & x > 4. \end{cases}$$

In analogy with the numerical test 1.3 pag. 481 in [**BCD97**] we compute the approximate solution obtaining the result in figure 3 and table in figure 1 (with considered control values $\{0, 1\}$) where from the exact solution we have $u_h(-1) = -2$, $u_h(0.5) = 0.75$ and $u_h(2) = 2$.

We remark that in this case for the interval $[0, 4]$, the Lipschitz constant of the function b is equal to the discount factor λ, so the equicontinuity estimate on the sequence u_h required in [**BCD97**], pag. 473 (1.6) does not hold.

n	$u_h^{(n)}(-1)$	$u_h^{(n)}(0.5)$	$u_h^{(n)}(2)$
1	1.6	1.95	2
2	1.24	1.8915	2
3	0.916	1.82792	2
4	0.6244	1.76183	2
5	0.36196	1.69518	2
6	0.125764	1.62938	2
7	−0.0868124	1.56546	2
8	−0.278131	1.50412	2
9	−0.450318	1.44582	2
10	−0.605286	1.39085	2
11	−0.744758	1.33935	2
12	−0.870282	1.293	2
13	−0.983254	1.25129	2
14	−1.08493	1.21374	2
15	−1.17644	1.17995	2
20	−1.51369	1.05542	2
30	−1.83044	0.938468	2
40	−1.94088	0.897688	2
50	−1.97938	0.883469	2
60	−1.99281	0.878511	2
70	−1.99749	0.876783	2
80	−1.99913	0.87618	2
90	−1.9997	0.87597	2
100	−1.99989	0.875896	2
120	−1.99999	0.875862	2
150	−2.	0.875857	2

TABLE 1

Next we test the algorithm with the discount factor $\lambda = 2$. In this case the estimate given by the theory can be applied and we can give for h small an error ≈ 0.37.

The exact solution of the control problem is:

$$u(x) = \begin{cases} \dfrac{x}{2} & x \leq 0 \\ \dfrac{x}{3} & 0 < x \leq 6 \\ 2 & x > 6. \end{cases}$$

and we test the numerical algorithm in $x = 0.5$ where $u(x) = 0.1666$, which belongs to the most significant interval $0 < x < 6$; in fact, in the other cases the approximate solution trivially converges quickly to the exact one.

n	1	2	3	4	5	6	7
$u_h^{(n)}(0.5)$	0.933333	0.548148	0.413169	0.366712	0.350902	0.345559	0.343762
n	8	9	10	11	12	13	14
	0.34316	0.342958	0.342891	0.342868	0.342861	0.342858	0.342858

⟨*code/MinPlusExample2.ma*⟩≡

⟨*The optimal control problem 2*⟩
```
 << KTL.ma
 << MinPlus.ma
```
⟨*Instatiation of templates with real number structure*⟩
⟨*Computation and visualization*⟩

In fact the optimal control problem allows to define the integral equation.

⟨*The optimal control problem 2*⟩≡
```
  (* Volterra Integral Equation aliasing an Optimal Control Problem *)
```
⟨*Definition of the kernel analog*⟩
⟨*Discretization Parameters 2*⟩
⟨*Optimal Control Problem definition 2*⟩

⟨*Discretization Parameters 2*⟩≡
```
  (* defintion of the state space *)
    oa = -2; ob = 6;
    Omega = {oa, ob}

  (* discretization parameters*)
    h = 7/10;
    k = 4/10;
    nk := nk = Floor[(Omega[[2]] - Omega[[1]]) / k ];
```

⟨*Optimal Control Problem definition 2*⟩≡
```
  (* set of control values *)
    Ac = {0, 1};

  (* problem dynamics *)
    b[x_, a_] := -a x;

  (* cost function *)
    f[x_, a_] := x;

  (* stopping cost function *)
    Psi[x_] := 2;

  (* discount factor *)
    Lambda = 1;
```

5.3. Application of the templates to usual integral equations. Now, we apply the templates to the approximation of the following integral equation of second kind:

$$u(x) = 1 + \int_0^x u(y)e^{3(x-y)}dy.$$

The exact solution of this equation is

$$u(x) = 1 + \int_0^x e^{4(x-y)}dy = \frac{3 + e^{4x}}{4}$$

and compared to the numerical runs we obtain the following table:

n	1	2	3	4	5	6
$u_h^{(n)}(1)$	12.0346	13.8941	14.4201	14.5357	14.5565	14.5597
$u_h^{(n)}(0.5)$	2.16929	2.53501	2.60403	2.6135	2.61453	2.61462

where the exact values are $u(1) = 14.3395$ and $u(0.5) = 2.5972$, and in the numerical test $k = 0.005$ and $\Omega = [0, 2]$.

⟨*code/KlassicExample.ma*⟩≡
⟨*Volterra Integral Equation*⟩
```
<< KTL.ma
<< Klassic.ma
```
⟨*Instatiation of templates with real number structure*⟩
⟨*Resolvent Approximation*⟩

⟨*Volterra Integral Equation*⟩≡
```
oa = 0 ; ob = 2;
Omega = {oa , ob};

G[x_,y_]:= Exp[3 ( x- y)];
Psi[x_] := 1 ;
lambdatilde = Lambda = 1 ;

IntegralDomain[n_,x_] := {0,x};
KernelDomain[n_, args_]:= {args[[1]],args[[2]]};

k=0.005;
nk=Floor[(Omega[[2]]-Omega[[1]]) / k];
```

⟨*Resolvent Approximation*⟩≡
```
tab = Table[{i,u[1],u[i,0.5]},{i,1,6}];
```

References

[BCD97] Martino Bardi and Italo Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, Systems & Control: Foundations & Applications, Birkhäuser Boston Inc., Boston, MA, 1997, With appendices by Maurizio Falcone and Pierpaolo Soravia. MR **99e:**49001

[CDI84] I. Capuzzo-Dolcetta and H. Ishii, *Approximate solutions of the Bellman equation of deterministic control theory*, Appl. Math. Optim. **11** (1984), no. 2, 161–181. MR **85f:**49046

[Fal87] M. Falcone, *A numerical approach to the infinite horizon problem of deterministic control theory*, Appl. Math. Optim. **15** (1987), no. 1, 1–13. MR **88c:**49025

[KM97] V.N. Kolokoltsov and V.P. Maslov, *Idempotent analysis and its applications. Appendix by Pierre Del Moral. Transl. from the Russian by V. E. Nazaikinskij.*, Mathematics and its Applications (Dordrecht). 401. Dordrecht: Kluwer Academic Publishers., 1997 (English).

[LM98] Grigori L. Litvinov and Victor P. Maslov, *The correspondence principle for idempotent calculus and some computer applications*, Idempotency (Bristol, 1994), Publ. Newton Inst., vol. 11, Cambridge Univ. Press, Cambridge, 1998, E-print math. GM/0101021 (http://arXiv.org), pp. 420–443. MR **99c:**16050

[LM00] G. L. Litvinov and E. V. Maslova, *Universal numerical algorithms and their software implementation*, Programmirovanie (2000), no. 5, 53–62, translation in Program. Comput. Software 26 (2000), no. 5, 275–280, E-print math. SC/0102114 (http://arXiv.org). MR 1 820 305

[LP01] P. Loreti and M. Pedicini, *Idempotent analogue of resolvent kernels for a deterministic optimal control problem*, Mat. Zametki **69** (2001), no. 2, 235–244. MR **2002d:**49038

[Mas87] V. P. Maslov, *On a new principle of superposition for optimization problems*, Uspekhi Mat. Nauk **42** (1987), no. 3(255), 39–48, 255. MR **88h:**35103

[MS92] V. P. Maslov and S. N. Samborskiĭ, *Stationary Hamilton-Jacobi and Bellman equations (existence and uniqueness of solutions)*, Idempotent analysis, Adv. Soviet Math., vol. 13, Amer. Math. Soc., Providence, RI, 1992, pp. 119–133. MR **94d:**49048

[VP36] V. Volterra and J. Pérès, *Théorie générale des fonctionelles*, Collection de monographies sur la théorie des fonctions, vol. 1, Gauthier-Villars, Paris, 1936.

Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate – Università degli Studi di Roma "La Sapienza", Via A. Scarpa 16, 00161 Roma

E-mail address: loreti@dmmm.uniroma1.it

Istituto per le Applicazioni del Calcolo "M. Picone" – Consiglio Nazionale delle Ricerche, Viale del Policlinico 137, 00161 Roma

E-mail address: marco@iac.cnr.it