A Patchy Dynamic Programming Method for the Numerical Solution of HJB Equations

M. Falcone

in collaboration with: S. Cacace, E. Cristiani, A. Picarelli

Dipartimento di Matematica

SAPIENZA Università di Roma

Sponsored by AFOSR Grant n. FA9550-10-1-0029 and

ITN SADCO

Initial Training Network

Sensitivity Analysis for Deterministic Controller Design

Workshop "Control and Optimization of PDEs"

Graz, October 12, 2011

Outline

Outline

Motivations

The numerical solution of optimal control problems via the Dynamic Programming approach is mainly motivated by the search for feedback controls for generic nonlinear Lipschitz continuous vectorfields and costs.

The solution of the corresponding Bellman equation in high dimension is a computationally intensive task and this bottleneck has limited the applications of this theory to industrial cases.

We want to overcome this technical problem developing new efficient algorithms with limited (and controlled) memory allocations and reasonable CPU times.

DP's advantages and disadvantages

PROS

1. The characterization of the value function is valid for all classical problems in any dimension.
2. The approximation is based on a-priori error estimates in L^∞, is valid in any dimension and does not require structured grids.
3. The computation of feedback controls is almost built in and there are nice results in low dimension.

CONS

The "curse of dimensionality" makes the problem difficult to solve in high dimension due to
1. computational cost
2. huge memory allocations.

DP's advantages and disadvantages

PROS

1. The characterization of the value function is valid for all classical problems in any dimension.
2. The approximation is based on a-priori error estimates in L^∞, is valid in any dimension and does not require structured grids.
3. The computation of feedback controls is almost built in and there are nice results in low dimension.

CONS

The "curse of dimensionality" makes the problem difficult to solve in high dimension due to

1. computational cost
2. huge memory allocations.

Tecnical difficulties

The bottleneck is the approximation of the value function v, however this remains the main goal since v allows to get back to feedback controls in a rather simple way.

For control problems

$$a^* \equiv \mathrm{argmin}[f(x,a) \cdot \nabla v(x) + l(x,a)]$$

For games

$$(a_e^*, a_p^*) \equiv \mathrm{argminmax}[f(x,a_e,a_p) \cdot \nabla v(x) + l(x,a_e,a_p)]$$

Control of PDEs via POD and HJB equations

POD decomposition allows to reduce the number of variables to approximate a partial differential equation.

The theory of viscosity solutions allows to characterize the value function as the unique weak solution of the HJB equation.

Our final goal is to approximate optimal control problems in infinite dimension coupling numerical schemes for HJBs with POD techniques.
Refs: Kunisch and Volkwein (2001, ...), Kunisch, Volkwein and Xie (2004)

Control of PDEs via POD and HJB equations

POD decomposition allows to reduce the number of variables to approximate a partial differential equation.

The theory of viscosity solutions allows to characterize the value function as the unique weak solution of the HJB equation.

Our final goal is to approximate optimal control problems in infinite dimension coupling numerical schemes for HJBs with POD techniques.

Refs: Kunisch and Volkwein (2001, ...), Kunisch, Volkwein and Xie (2004)

Control of PDEs via POD and HJB equations

POD decomposition allows to reduce the number of variables to approximate a partial differential equation.

The theory of viscosity solutions allows to characterize the value function as the unique weak solution of the HJB equation.

Our final goal is to approximate optimal control problems in infinite dimension coupling numerical schemes for HJBs with POD techniques.
Refs: Kunisch and Volkwein (2001, ...), Kunisch, Volkwein and Xie (2004)

Outline

Proper Orthogonal Decomposition and SVD

Given $y_1, \ldots, y_n \in \mathbb{R}^m$, let $V = \text{span}\{y_1, \ldots, y_n\} \subset \mathbb{R}^m$

We look for an orthonormal basis $\{\psi_i\}_{i=1}^{\ell}$ in $\mathbb{R}^m$ with $\ell \leq \dim V$ such that

$$J(\psi_1, \ldots, \psi_\ell) = \sum_{j=1}^{n} \left\| y_j - \sum_{i=1}^{\ell} < y_j, \psi_i > \psi_i \right\|$$

reaches a minimum.

Constrained Problem

$$\min J(\psi_1, \ldots, \psi_\ell) \quad \text{subject to} < \psi_i, \psi_j >= \delta_{ij}$$

Proper Orthogonal Decomposition and SVD

Given $y_1, \ldots, y_n \in \mathbb{R}^m$, let $V = \text{span}\{y_1, \ldots, y_n\} \subset \mathbb{R}^m$

We look for an orthonormal basis $\{\psi_i\}_{i=1}^{\ell}$ in $\mathbb{R}^m$ with $\ell \leq \dim V$ such that

$$J(\psi_1, \ldots, \psi_\ell) = \sum_{j=1}^{n} \left\| y_j - \sum_{i=1}^{\ell} < y_j, \psi_i > \psi_i \right\|$$

reaches a minimum.

Constrained Problem

$$\min J(\psi_1, \ldots, \psi_\ell) \quad \text{subject to} < \psi_i, \psi_j >= \delta_{ij}$$

Proper Orthogonal Decomposition and SVD

Given $y_1, \dots, y_n \in \mathbb{R}^m$, let $V = \text{span}\{y_1, \dots, y_n\} \subset \mathbb{R}^m$

We look for an orthonormal basis $\{\psi_i\}_{i=1}^{\ell}$ in $\mathbb{R}^m$ with $\ell \leq \dim V$ such that

$$J(\psi_1, \dots, \psi_\ell) = \sum_{j=1}^{n} \left\| y_j - \sum_{i=1}^{\ell} < y_j, \psi_i > \psi_i \right\|$$

reaches a minimum.

Constrained Problem

$$\min J(\psi_1, \dots, \psi_\ell) \quad \text{subject to} < \psi_i, \psi_j >= \delta_{ij}$$

Theorem (Kunisch, Volkwein)

Let $Y = [y_1, \dots, y_n] \in \mathbb{R}^{m\times n}$ be a given matrix with rank $d \le \min\{m, n\}$.
Further, let $Y = \Psi\Sigma V^T$ be the SVD of Y, where
$\Psi = [\psi_1, \dots, \psi_m] \in \mathbb{R}^{m\times m}$, $V = [v_1, \dots, v_n] \in \mathbb{R}^{n\times n}$ are orthogonal matrices and the matrix $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal.
Then, for any $\ell \in \{1, \dots, d\}$ the solution to

$$\min J(\psi_1, \dots, \psi_\ell) = \sum_{j=1}^{n} \left\| y_j - \sum_{i=1}^{\ell} < y_j, \psi_i > \psi_i \right\|$$

$$\text{such that } < \psi_i, \psi_j >= \delta_{ij} \text{ per } 1 \le i, j \le \ell$$

is given by the singular vectors $\{\psi_i\}_{i=1}^{\ell}$, i.e, by the first ℓ columns of Ψ.

Definition

For $\ell \in \{1, \dots, d\}$, the vectors $\{\psi_i\}_{i=1}^{\ell}$ are called *POD basis* of rank ℓ.

Computation of POD basis

If $n < m$

$$YY^T v_i = \lambda_i v_i \qquad \text{for } i = 1, \dots, \ell$$

and setting $\psi_i = \dfrac{1}{\sqrt{\lambda_i}} Y v_i$.

Reference

S. Volkwein. *Model Reduction using Proper Orthogonal Decomposition*, 2007 http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Vorlesung.pdf

Definition

For $\ell \in \{1, \ldots, d\}$, the vectors $\{\psi_i\}_{i=1}^{\ell}$ are called *POD basis* of rank ℓ.

Computation of POD basis

If $n < m$

$$YY^T v_i = \lambda_i v_i \qquad \text{for } i = 1, \ldots, \ell$$

and setting $\psi_i = \dfrac{1}{\sqrt{\lambda_i}} Y v_i$.

Reference

S. Volkwein. *Model Reduction using Proper Orthogonal Decomposition*, 2007 http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Vorlesung.pdf

Definition

For $\ell \in \{1, \dots, d\}$, the vectors $\{\psi_i\}_{i=1}^{\ell}$ are called *POD basis* of rank ℓ.

Computation of POD basis

If $n < m$

$$YY^T v_i = \lambda_i v_i \qquad \text{for } i = 1, \dots, \ell$$

and setting $\psi_i = \dfrac{1}{\sqrt{\lambda_i}} Y v_i$.

Reference

S. Volkwein. *Model Reduction using Proper Orthogonal Decomposition*, 2007 http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Vorlesung.pdf

A typical application

Let us consider the following ODEs system:

$$\begin{cases} \dot{y}(t) = Ay(t) + f(t, y(t)), \;\; t \in (0, T] \\ \\ y(0) = y_0 \end{cases} \tag{1}$$

where $y_0 \in \mathbb{R}^m, A \in \mathbb{R}^{m \times m}$ and $f : [0, T] \times \mathbb{R}^m \to \mathbb{R}^m$ is continuous and locally Lipschitz.
Let us suppose to know that the solution at each time t_j, with $0 \leq t_1 \leq \ldots \leq t_n \leq T$, verifies

Snapshots

$$y_j = y(t_j) = e^{t_j A} y_0 + \int_0^{t_j} e^{(t_j - s)A} f(s, y(s))\, ds \quad j \in \{1, \ldots, n\},$$

Let $\{\psi_j\}_{j=1}^{\ell}$ be a POD basis, we make the following ansatz:

$$y^{\ell}(t) = \sum_{j=1}^{\ell} y_j^{\ell}(t)\psi_j = \sum_{j=1}^{\ell} < y^{\ell}(t), \psi_j > \psi_j, \qquad \forall t \in [0, T]$$

Reduced-Order Modelling

$$\begin{cases} \sum_{j=1}^{\ell} \dot{y}_j^{\ell}(t)\psi_j = \sum_{j=1}^{\ell} y_j^{\ell}(t)A\psi_j + f(t, y^{\ell}(t)), & t \in (0, T] \\ \\ \sum_{j=1}^{\ell} y_j^{\ell}(0)\psi_j = y_0. \end{cases}$$

Let $\{\psi_j\}_{j=1}^{\ell}$ be a POD basis, we make the following ansatz:

$$y^\ell(t) = \sum_{j=1}^{\ell} y_j^\ell(t)\psi_j = \sum_{j=1}^{\ell} < y^\ell(t), \psi_j > \psi_j, \qquad \forall t \in [0, T]$$

Reduced-Order Modelling

$$\begin{cases} \sum_{j=1}^{\ell} \dot{y}_j^\ell(t)\psi_j = \sum_{j=1}^{\ell} y_j^\ell(t)A\psi_j + f(t, y^\ell(t)), & t \in (0, T] \\ \sum_{j=1}^{\ell} y_j^\ell(0)\psi_j = y_0. \end{cases}$$

From the reduced model, it follows

$$\dot{y}_i^\ell(t) = \sum_{j=1}^{\ell} y_j^\ell(t) < A\psi_j, \psi_i > + < f(t, y^\ell(t)), \psi_i >$$

and with compact notations:

$$\begin{cases} \dot{y}^\ell(t) = A^\ell y^\ell(t) + F(t, y^\ell(t)) \\ \\ y^\ell(0) = y_0^\ell \end{cases}$$

where:

$$A^\ell \in \mathbb{R}^{\ell \times \ell} \qquad \text{with } (A^\ell)_{ij} =< A\psi_i, \psi_j >,$$

$$y^\ell = \begin{pmatrix} y_1^\ell \\ \vdots \\ y_\ell^\ell \end{pmatrix} : [0, T] \to \mathbb{R}^\ell$$

$$F = (F_1, \dots, F_\ell)^T : [0, T] \times \mathbb{R}^\ell \to \mathbb{R}^\ell$$

$$F_i(t, y) = \left\langle f\left(t, \sum_{j=1}^{\ell} y_j \psi_j\right), \psi_i \right\rangle \text{ for } t \in [0, T] \;\; y = (y_1, \dots y_\ell) \in \mathbb{R}^\ell.$$

$$y_0^\ell = \begin{pmatrix} < y_0, \psi_1 > \\ \vdots \\ < y_0, \psi_\ell > \end{pmatrix} \in \mathbb{R}^\ell.$$

Remark

We obtain system of ODEs approximating evolutive PDEs by finite differences or finite elements schemes.

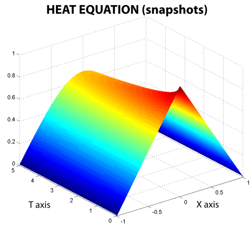
TEST 1

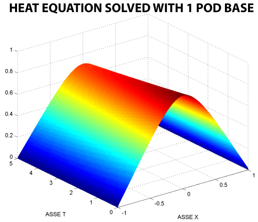
A Parabolic Problem

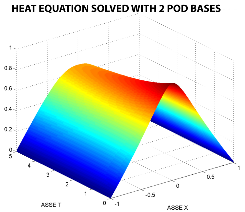
$$\begin{cases} \dfrac{d}{dt}y(x,t) = \dfrac{1}{60}\dfrac{d^2}{dx^2}y(x,t), \quad x \in [-1,1], \; t \in (0,5] \\ \\ y(-1,t) = y(1,t) = 0, \qquad y(x,0) = 1 - |x| \end{cases}$$

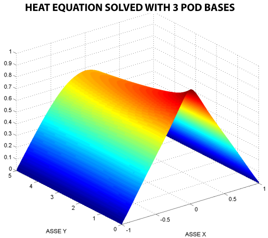
Snapshots Parameters

$\Delta x = 0.02, \Delta t = 0.012, Nr = 100$


HEAT EQUATION (snapshots)
T axis
X axis


HEAT EQUATION SOLVED WITH 1 POD BASE
ASSE T
ASSE X


HEAT EQUATION SOLVED WITH 2 POD BASES
ASSE T
ASSE X


HEAT EQUATION SOLVED WITH 3 POD BASES
ASSE Y
ASSE X

TEST 2

An Hyperbolic Problem

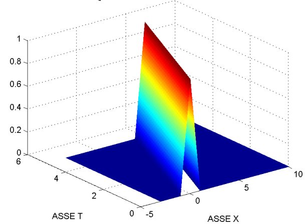
$$\begin{cases} \dfrac{d}{dt}y(x,t) + \dfrac{d}{dx}y(x,t) = 0, \quad x \in \mathbb{R}, \;\; t \in (0, T], \\ \\ y(x,0) = \max\{1 - |x|, 0\}, \end{cases}$$

Snapshots Parameters

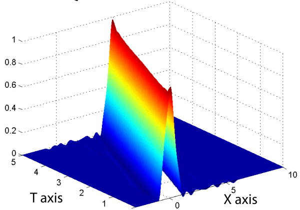
$\Delta x = 0.01, \Delta t = 0.01, Nr = 1400$

TEST 2

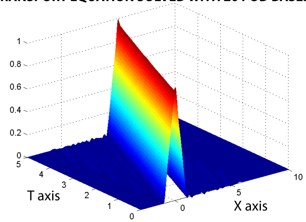
TRANSPORT EQUATION - ANALYTIC SOLUTION



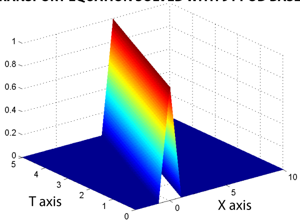
TRANSPORT EQUATION SOLVED WITH 11 POD BASES



TRANSPORT EQUATION SOLVED WITH 20 POD BASES



TRANSPORT EQUATION SOLVED WITH 91 POD BASES



POD	$\mathcal{E}(\ell)$	L^1	L^2
1	0.9661	0.0647	0.0554
2	0.9941	0.0164	0.0156
3	0.9983	0.0062	0.0062
5	0.9996	0.0015	0.0016
20	1	1.9493e-004	0.0014

Table: L^1 and L^2 errors for TEST 1 (parabolic)

POD	$\mathcal{E}(\ell)$	L^1	L^2
11	0.9082	0.1150	0.0636
20	0.9511	0.0442	0.0258
91	0.9901	0.0028	0.0022

Table: L^1 and L^2 errors for TEST 2 (hyperbolic)

Patchy decomposition

Main Idea

Since the patches are invariant with respect to the patchy vector fields, we can split the computation of the solution in D sub-problems, each corresponding to a patchy sub-domain and use a parallel algorithm to compute the value function in the whole domain.

This patchy domain decomposition method has shown to be more efficient with respect to standard (static) domain decomposition techniques as we will show by some numerical tests.

Previous works based on Al'brekht series expansion by Krener and Navasca (2007,..), Hunt (PhD thesis, 2011).

Outline

The model problem

Let us consider, for example, the infinite horizon optimal control problem which leads to the Hamilton-Jacobi-Bellman equation

$$\lambda v(x) + \max_{a \in A} \{-f(x,a) \cdot \nabla v(x) - l(x,a)\} = 0\,, \qquad x \in \Omega$$

where A is a compact subset of $\mathbb{R}^m$ and f, l are given functions, $\lambda > 0$.

The infinite horizon problem

Dynamics

$$\begin{cases} \dot{y}(t) = f(y(t), \alpha(t)) & t > 0 \\ y(0) = x \end{cases}$$

Admissible controls

$$\alpha(\cdot) \in \mathcal{A} \equiv \{\alpha : [0, +\infty[\to A, \mathrm{measurable}\}$$

Cost

$$J(x, \alpha(\cdot)) = \int_0^\infty l(y(t), \alpha(t)) e^{-\lambda t} dt$$

Value function

$$v(x) = \inf_{\alpha(\cdot) \in \mathcal{A}} J(x, \alpha(\cdot))$$

The infinite horizon problem

For numerical purposes we have to deal the problem in a bounded domain Ω

$$\lambda v(x) + \max_{u \in U} \{-f(x,u) \cdot \nabla v(x) - l(x,u)\} = 0\,, \qquad x \in \Omega$$

Domain splitting

Let us consider a splitting of Ω into D subdomains $\Omega_d, d = 1, \ldots, D$

$$\Omega = \cup_d \Omega_d$$

and a grid G with a number of nodes N_Ω

$$N_\Omega \approx N_1 + \ldots + N_D$$

Sketch of Classical DD

MAIN CICLE

REPEAT

STEP 1

Compute one iteration of the numerical operator S restricted to every domain Ω_d, $d = 1, \ldots, D$

STEP 2

Couple the information on the overlapping zones (Transmission Conditions)

UNTIL a stopping rule is satisfied

Sketch of Classical DD

MAIN CICLE

REPEAT

STEP 1

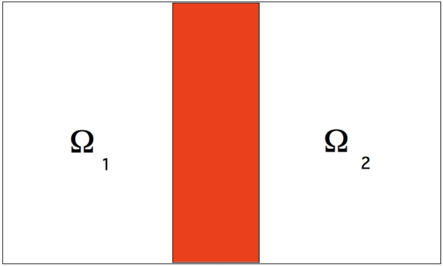
Compute one iteration of the numerical operator S restricted to every domain Ω_d, $d = 1, \ldots, D$

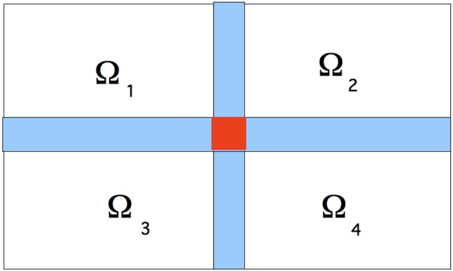
STEP 2

Couple the information on the overlapping zones (Transmission Conditions)

UNTIL a stopping rule is satisfied

Transmission Conditions





The correct transmission condition is the min operator

$$\min\{S_1, S_2, \ldots, S_D\}.$$

Outline

Main goal

We want to construct a domain decomposition which is based on the patches defined by Ancona and Bressan.
PROS patches are invariant with respect to the optimal dynamics
CONS we need a dynamic construction of the patches.

Let be $\Omega \subset \mathbb{R}^n$ an open domain with smooth boundary $\partial\Omega$ and g a smooth vector field defined on a neighborhood of $\overline{\Omega}$.

Definition

We say that the pair (Ω, g) is a patch if Ω is a positive-invariant region for g, i.e. at every boundary point $x \in \partial\Omega$ the inner product of g with the outer normal n satisfies

$$\langle g(x), n(x) \rangle < 0.$$

Patchy vectorfields

A patchy vector field on a domain $\Omega \subset \mathbb{R}^n$ is a superposition of patches, as reported in the following

Definition

We say that $g : \Omega \to \mathbb{R}^n$ is a patchy vector field if there exists a family of patches $\{(\Omega_\alpha, g_\alpha) : \alpha \in \mathcal{I}\}$ such that

- $\mathcal{I}$ is a totally ordered index set,
- the open sets Ω_α form a locally finite covering of Ω,
- the vector field g can be written in the form

$$g(x) = g_\alpha(x) \qquad \text{if} \qquad x \in \Omega_\alpha \setminus \bigcup_{\beta > \alpha} \Omega_\beta .$$

Outline

Goal

To build a domain decomposition such that

- the solution in each patch does not depend on the solution in other patches;
- there is no transmission condition through the boundaries of the patches.

In this way the computation can be fully parallelized. The final solution is obtained by merging all the patches at the end.

To this end we need an a-priori knowledge of characteristics which is not available $\Rightarrow$ PRE-COMPUTATIONS

The Patchy Algorithm

Step1. (Computation on the coarse grid). We solve the equation on a coarse grid G_{coarse} by means of the classical domain decomposition technique. This leads to the function u_{coarse}.

Step2. (Interpolation on a fine grid). We compute a first approximation $u^{(0)}$ of the solution on a fine grid G_{fine} by means of a simple bilinear interpolation of the values u_{coarse}.
We also compute the optimal control

$$a^*_{\text{coarse}}(x_i) = \arg\max_a \{-f(x_i, a) \cdot \nabla u^{(0)}(x_i)\}, \quad x_i \in G_{\text{fine}}$$

Note that a^*_{coarse} is defined on G_{fine}.

The Patchy Algorithm

Step3. (Partition of target). On G_{fine}, we divide the target in N_p parts denoted by Ω_0^j, $j = 1, \ldots, N_p$.

Step4. (Main cycle) For any $j = 1, \ldots, N_p$,

- Step4.1. (Creation of j-th patch). We use the (coarse) optimal control a^*_{coarse} to find the nodes of the grid G_{fine} which have Ω_0^j in their numerical domain of dependence. This procedure defines the j-th patch. (NEXT SLIDE)
- Step4.2. (Computation in patches). We solve iteratively the equation in the j-th patch until convergence is reached, imposing state constraints boundary conditions.

Step5. (Merge). All the solutions computed in the N_p patches are assembled in the grid G_{fine}.

The Patchy Algorithm: creation of a patch

For $j = 1, \dots, N_p$

1. (Initialization) Set

$$\phi_i = \begin{cases} 1\,, & x_i \in \Omega_0^j \\ 0\,, & x_i \in G_{\text{fine}} \backslash \Omega_0^j \end{cases}\,, \qquad i = 1, \dots, N.$$

2. (Iteration) Solve iteratively the following *ad hoc* numerical scheme, until convergence is reached.

$$\phi_i = \phi(x_i + h_i f(x_i, a^*_{\text{coarse}}(x_i)))\,, \qquad i = 1, \dots, N.$$

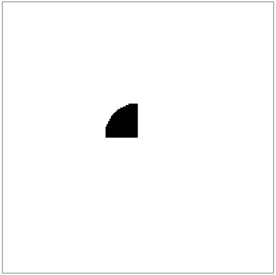
   Note that the solution ϕ takes values in $[0, 1]$.

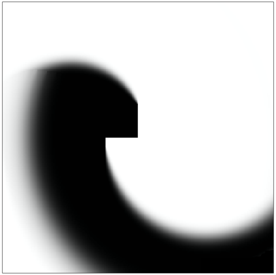
3. (Projection) Project the color j into a binary value

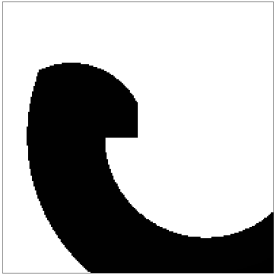
$$\phi_i = \begin{cases} 1\,, & \phi(x_i) \geq \frac{1}{2} \\ 0\,, & \phi(x_i) < \frac{1}{2} \end{cases}\,, \qquad i = 1, \dots, N.$$

   The sub-domain $P_j = \{x_i : \phi(x_i) = 1\}$ is the j-th patch.

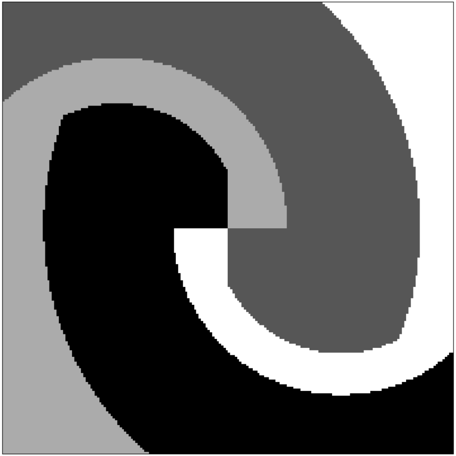
The Patchy Algorithm: creation of a patch


Initialization


Iteration


Projection

The Patchy Algorithm: invariant domain decomposition

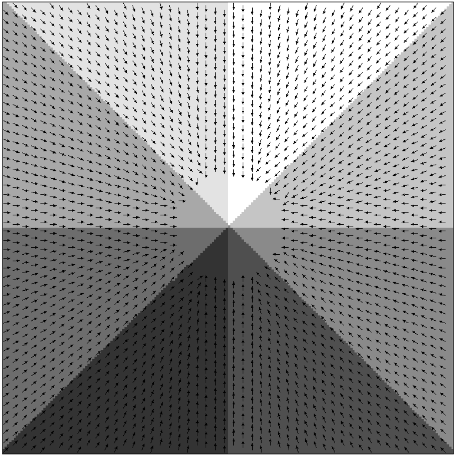


Patchy domain decomposition (4 patches)

Numerical Tests in dimension 2

Test 1: Eikonal

$$f(x_1, x_2, a) = a\,, \qquad A = B(0,1)\,, \qquad \Omega_0 = B(0, 0.5).$$



Patchy domain decomposition (8 patches)

Test 1: Eikonal – Patchy Error

We compute the difference between the patchy solution and the DD solution. Since the scheme is the same, this error is due to the fact that patches are not perfectly independent.

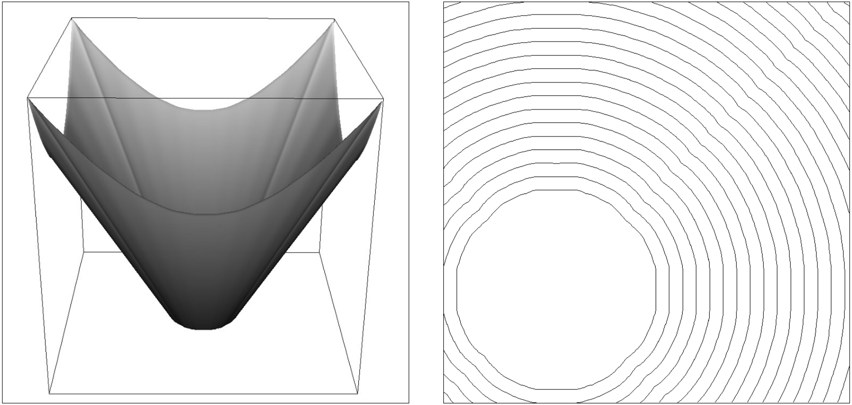
Error table in norm $\|\cdot\|_1$ ($\|\cdot\|_\infty$) depending on the space steps k_{coarse} and k_{fine}.

	k_f=0.08	k_f=0.04	k_f=0.02	k_f=0.01	k_f=0.005
k_c=0.08	0.436 (0.960)	0.275 (1.856)	0.102 (0.048)	0.065 (0.034)	0.048 (0.026)
k_c=0.04	–	0.088 (0.046)	0.029 (0.023)	0.014 (0.042)	0.005 (0.008)
k_c=0.02	–	–	0.038 (0.029)	0.012 (0.013)	0.004 (0.008)
k_c=0.01	–	–	–	0.011 (0.016)	0.006 (0.010)
k_c=0.005	–	–	–	–	0.004 (0.008)

$A = B(0,1)$ is discretized with 32 points and the number of patches is 16.

Test 1: Eikonal

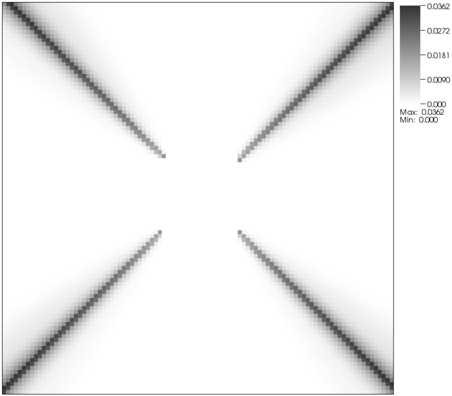
Patchy solution



The subsolutions merge quite well!

Test 1: Eikonal

Patchy error



The error is localized on the boundaries of the patches!

Patchy method vs classical Domain Decomposition

CPU times (in seconds) depending on the number of processors and the number of patches

Controls: 16. Grid: $100^2 \longrightarrow 800^2$

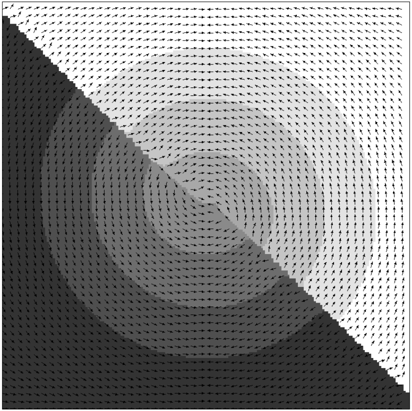
	2 domains	4 domains	8 domains	16 domains	Best DD
1 proc	1547	1076	1058	933	1571
2 procs	845	595	574	504	820
4 procs	459	325	317	271	415

Controls: 32. Grid: $100^2 \longrightarrow 800^2$

	2 domains	4 domains	8 domains	16 domains	Best DD
1 proc	2702	1897	1843	1623	2785
2 procs	1462	998	968	872	1430
4 procs	771	532	514	435	716

Test 2: Fan

$$f(x_1, x_2, a) = |x_1 + x_2 + 0.1|a\,, \qquad A = B(0,1)\,, \qquad \Omega_0 = \{x_1 = 0\}.$$



Patchy domain decomposition (8 patches)

Test 2: Fan

Patchy error

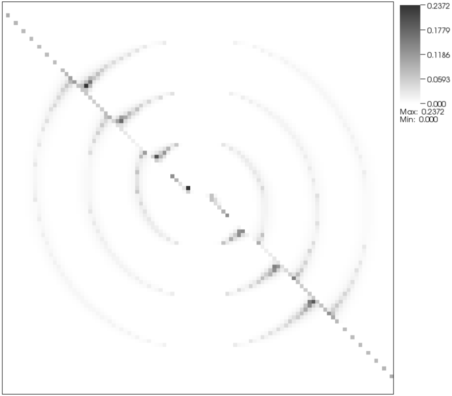
Error table in norm $\|\cdot\|_1$ ($\|\cdot\|_\infty$) depending on the space steps k_{coarse} and k_{fine}.

	$k_f = 0.08$	$k_f = 0.04$	$k_f = 0.02$	$k_f = 0.01$	$k_f = 0.005$
k_c=0.08	1.393 (3.023)	0.123 (1.507)	0.037 (0.315)	0.017 (0.263)	0.011 (0.263)
k_c=0.04	–	0.114 (1.502)	0.032 (0.149)	0.011 (0.095)	0.006 (0.095)
k_c=0.02	–	–	0.032 (0.111)	0.011 (0.061)	0.004 (0.037)
k_c=0.01	–	–	–	0.011 (0.079)	0.004 (0.037)
k_c=0.005	–	–	–	–	0.004 (0.037)

$A = B(0, 1)$ is discretized with 32 points and the number of patches is 16.

Test 2: Fan

Patchy Error



The error is localized on the boundaries of the patches!

Test 2: Fan

Patchy method vs classical Domain Decomposition

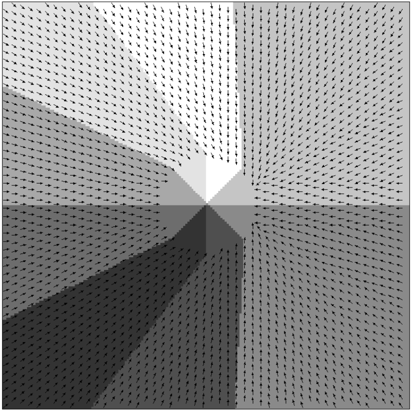
CPU times (in seconds) depending on the number of processors and the number of patches

Controls: 32. Grid: $100^2 \to 800^2$

	2 domains	4 domains	8 domains	16 domains	Best DD
1 proc	3712	3322	3049	3172	4163
2 procs	2020	1746	1596	1559	2124
4 procs	1032	900	841	852	1069

Test 3: Zermelo

$$f(x_1, x_2, a) = 2.1a + (2,0)\,, \qquad A = B(0,1)\,, \qquad \Omega_0 = B(0, 0.5).$$



Patchy domain decomposition (8 patches)

Patchy Error

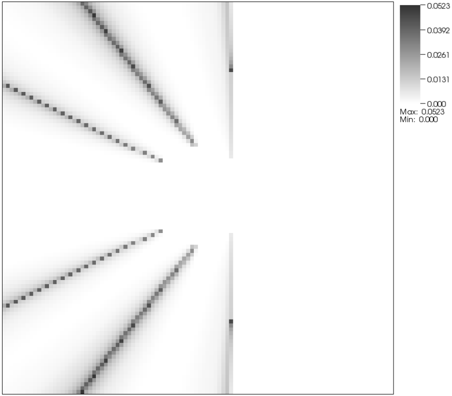
Error table in norm $\|\cdot\|_1$ ($\|\cdot\|_\infty$) depending on the space steps k_{coarse} and k_{fine}.

	$k_f = 0.08$	$k_f = 0.04$	$k_f = 0.02$	$k_f = 0.01$	$k_f = 0.005$
k_c=0.08	0.171 (0.293)	0.159 (0.059)	0.097 (0.057)	0.026 (0.027)	0.006 (0.016)
k_c=0.04	–	0.101 (0.063)	0.033 (0.041)	0.011 (0.023)	0.004 (0.016)
k_c=0.02	–	–	0.039 (0.039)	0.012 (0.023)	0.004 (0.016)
k_c=0.01	–	–	–	0.011 (0.020)	0.005 (0.015)
k_c=0.005	–	–	–	–	0.004 (0.016)

$A = B(0,1)$ is discretized with 32 points and the number of patches is 16.

Test 3: Zermelo

Patchy Error



The error is localized on the boundaries of the patches!

Patchy method vs classical Domain Decomposition

Cpu times (in seconds) depending on the number of processors and the number of patches

Controls: 32. Grid: $100^2 \longrightarrow 800^2$

	2 domains	4 domains	8 domains	16 domains	Best DD
1 proc	3113	2675	2126	2018	3209
2 procs	1651	1404	1111	1054	1640
4 procs	871	721	584	545	825

Numerical Tests in dimension 3

Here several add-on's are enabled!
(ordering of the nodes (FMM-like), reduced controls, ...)

Test 1: Eikonal

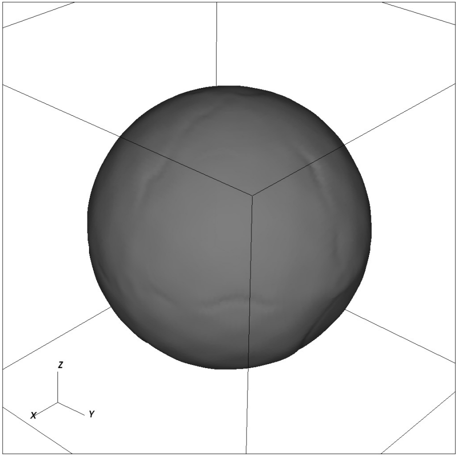
$$f(x_1, x_2, x_3, a) = a\,, \quad A = B(0,1)\,, \quad \Omega_0 = B(0, 0.5).$$

dynamics	grid size	CPU time	Error L^1	Error L^∞
Eikonal 3D	$50^3 \to 100^3$	183	0.033	0.035
Eikonal 3D	$50^3 \to 200^3$	1217	0.029	0.042

$A = B(0,1)$ is discretized with 189 points (then reduced when working on the fine grid) and the number of patches is 8. Processors are 4.

Test 1: Eikonal

One level set of the patchy solution



The error is localized on the boundaries of the patches!

Test 2: Fan

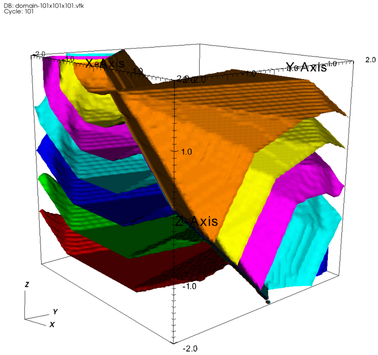
$$f(x_1, x_2, x_3, a) = |x_1 + x_2 + x_3 + 0.1|a\,, \quad A = B(0,1)\,, \quad \Omega_0 = \{x_1 = 0\}.$$

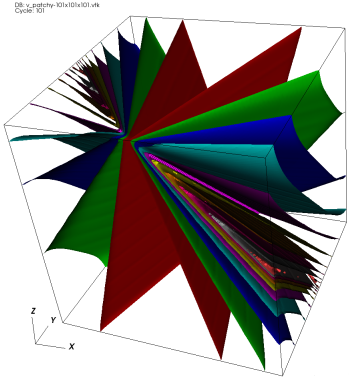
dynamics	grid size	CPU time	Error L^1	Error L^∞
Fan 3D	$50^3 \rightarrow 100^3$	165	0.064	0.187
Fan 3D	$50^3 \rightarrow 200^3$	1269	0.056	0.305

$A = B(0,1)$ is discretized with 189 points (then reduced when working on the fine grid) and the number of patches is 8. Processors are 4.

Test 2: Fan

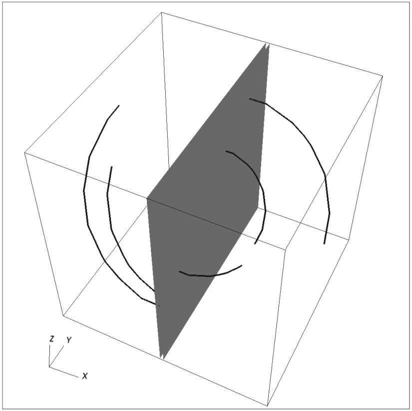
Patchy domain decomposition (8 patches) and level sets of the patchy solution





Test 2: Fan

Some optimal trajectories to the target



Conclusions and future directions

We developed an approximation method for the solution of Hamilton-Jacobi equations which combines a patchy decomposition of the domain and a dynamic programming scheme.

The method can handle:

- more general control problems (minimum time, finite horizon, ...)
- state constraints
- pursuit evasion games

The numerical tests show a very small and localized error (on the patches boundaries).

Future directions

We want to analyze the method (convergence, error estimates) and combine this technique with efficient fast marching techniques.

Efficient coupling of this method with POD techniques for the control of PDEs (ongoing with A. Alla).

References

- M. Falcone, R. Ferretti, Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations , SIAM, book in preparation
- S. Cacace, E. Cristiani, M. Falcone, A. Picarelli, *A patchy domain decomposition method for a class of Hamilton-Jacobi-Bellman equations*, preprint 2011, submitted to SIAM J. Sci. Comp. available at `http://arxiv.org/pdf/1109.3577v1`
- M. Falcone, P. Lanucara, and A. Seghini, *A splitting algorithm for Hamilton-Jacobi-Bellman equations*, Applied Numerical Mathematics, 15 (1994), pp. 207-218.