Interval Based
Parallel Computing
of the Viability Kernel

Stéphane Le Ménec
MBDA
Le Plessis-Robinson

SWIM 2016
Lyon

Work performs with support of :

ANR ASTRID Maturation VIATIC2
And PIA CAPACITES

Overview

1. **Viability**

Viability Kernel

Capture Basin

Viability : a Tool for Autonomous Decision Making

2. **Interval Arithmetic**

Set Inversion Via Interval Analysis

Contractor Programming

3. **Interval Based Viability Algorithms**

Bisection Based Approach

Differential Inclusion Contractors

4. **Parallel Computing**

Many Core Computing Architecture

On-board Computing Constraints

5. **Numerical Results**

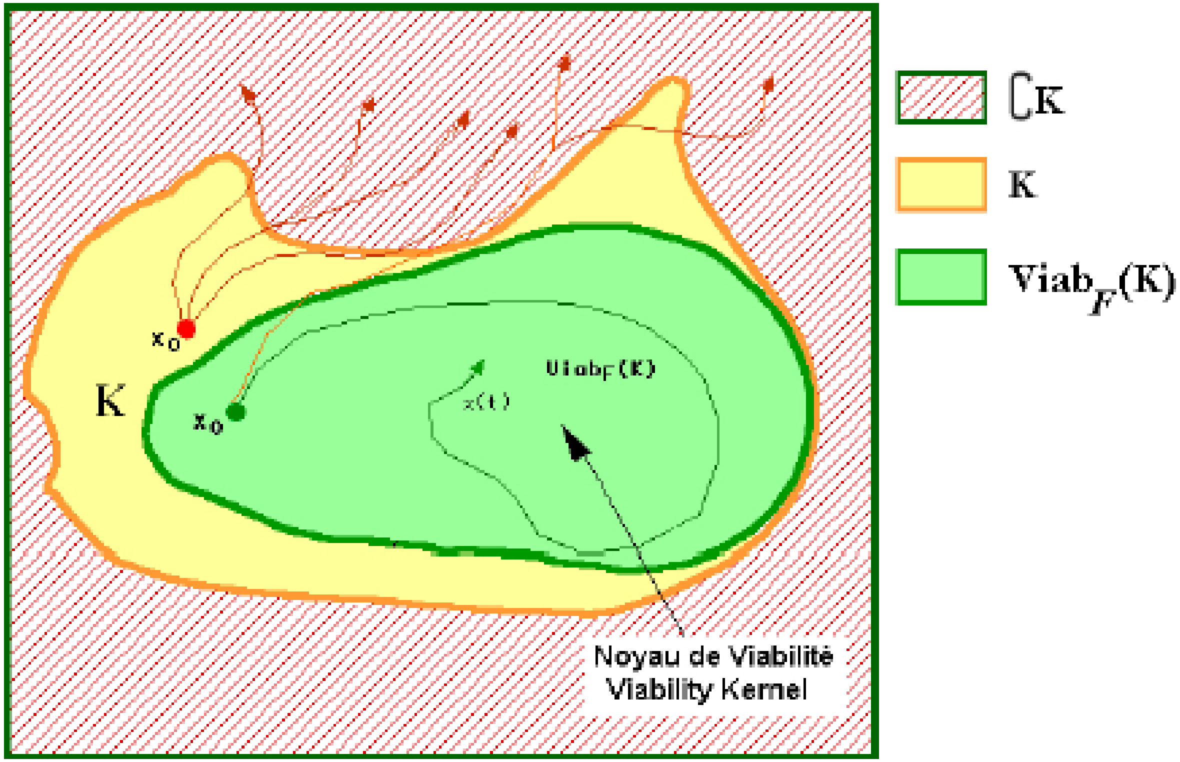
Benchmark Example : Car On The Hill (COTH) Problem

Mono-Core and Multi-Cores Results

Computing Time Performances

6. **Synthesis, Conclusion and Way Forward**

Viability Kernel

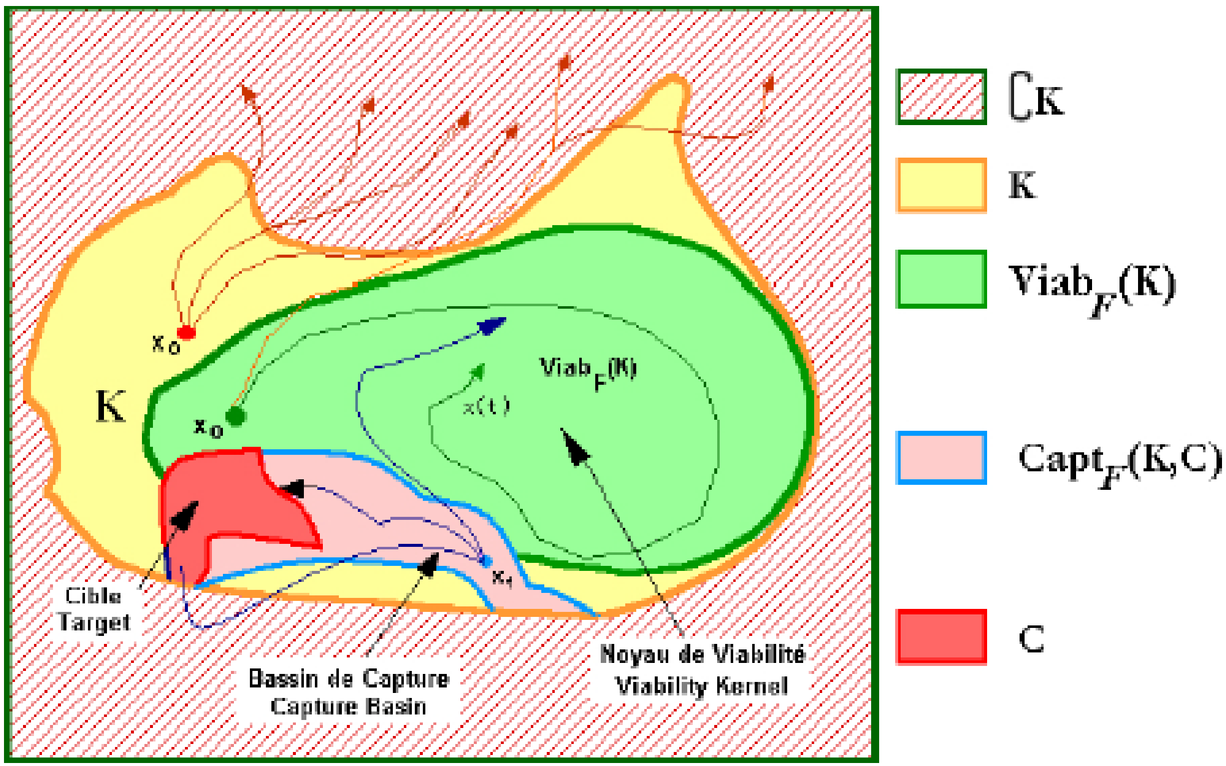


2 Viability Kernel.
It is the set of the states in the environment K from which starts *at least* one evolution that remains always in K . All viable evolution remains necessarily always in $Viab_F(K)$.

$$Viab_F(K) := \{x \in K \mid \exists x(\cdot) \in \mathcal{S}_F(x),\ \forall t > 0,\ x(t) \in K\}$$

For more details about Viability Theory, please refer to Professor Jean-Pierre Aubin and co-authors

Capture Basin



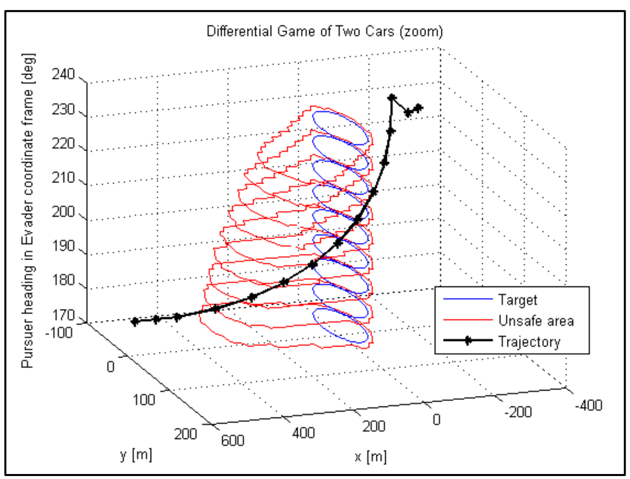
5 **Capture Basin.**
The basin of capture of the target, viable in an environment, is the subset (possibly empty) of the states of the environment K from which *at least* one evolution remains *viable* in the environment until it reaches the target *in finite time*.

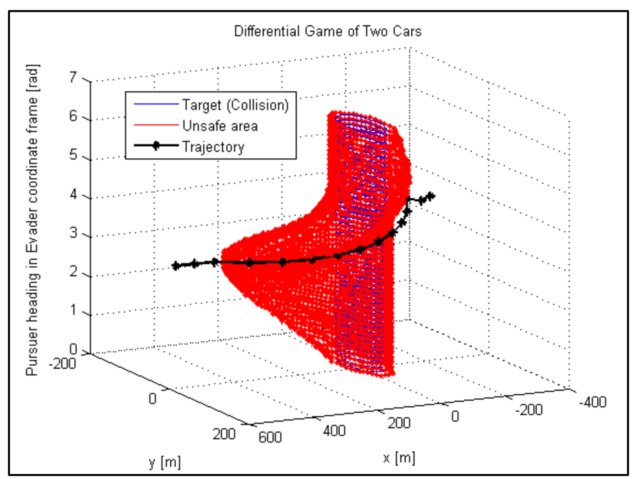
$$Capt_F(K,C) := \{x \in K \mid \exists x(\cdot) \in \mathcal{S}_F(x), \exists t^* > 0,$$
$$x(t^*) \in C,\ \forall t \in [0,t^*],\ x(t) \in K\}$$

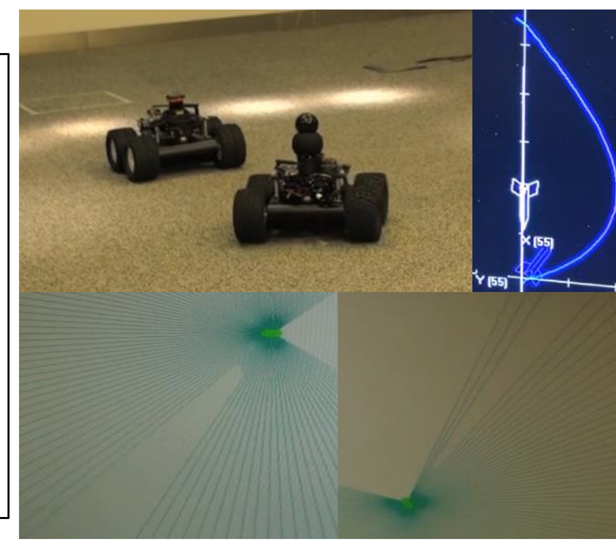
Viability : A Tool for Autonomous Decision Making

Viability Theory addresses question marks that happen in autonomous systems :

- Validation & Verification (safety) of complex systems
- Vehicle attainability (computation of forward and backward reachable sets)
- Collision avoidance (of fix or moving obstacles)
- Can handle various types of uncertainties (environment, model, controls, external agents)
- Can consider dynamics with hybrid behaviours : switches resulting from decisions, controls, kinematics changes …





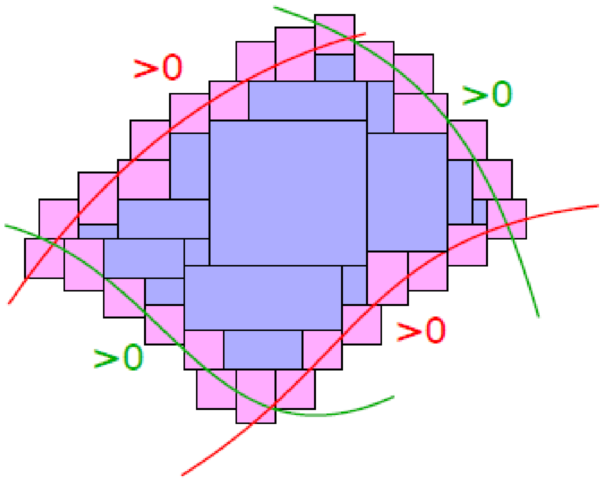


Interval Arithmetics

Interval computation allows solving in an easy manner such problems:

Consider $f : \mathbb{R}^n \to \mathbb{R}^m$.

$$\mathcal{S}_1 \cup \mathcal{S}_2 \supseteq \{x \in \mathbb{R}^n,\ \forall i,\ 1 \leq i \leq m,\ f_i(x) \leq 0\} \supseteq \mathcal{S}_1$$



➡ Set Inversion Via Interval Analysis (SIVIA) algorithms

(Based on bisection and contractor techniques)

For more details about interval arithmetic, please refer to Professor Luc Jaulin and co-authors

IBEX

- IBEX is a C++ library for interval computation allowing to :
- Perform simulations (guaranteed integration) in the context of intervals
- Implement contractors :

The operator $\mathcal{C}_{\mathbb{X}} : \mathbb{IR}^n \to \mathbb{IR}^n$ is a *contractor* for $\mathbb{X} \subset \mathbb{R}^n$ if

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \begin{cases} \mathcal{C}_{\mathbb{X}}([\mathbf{x}]) \subset [\mathbf{x}] & \text{(contractance)}, \\ \mathcal{C}_{\mathbb{X}}([\mathbf{x}]) \cap \mathbb{X} = [\mathbf{x}] \cap \mathbb{X} & \text{(completeness)}. \end{cases}$$

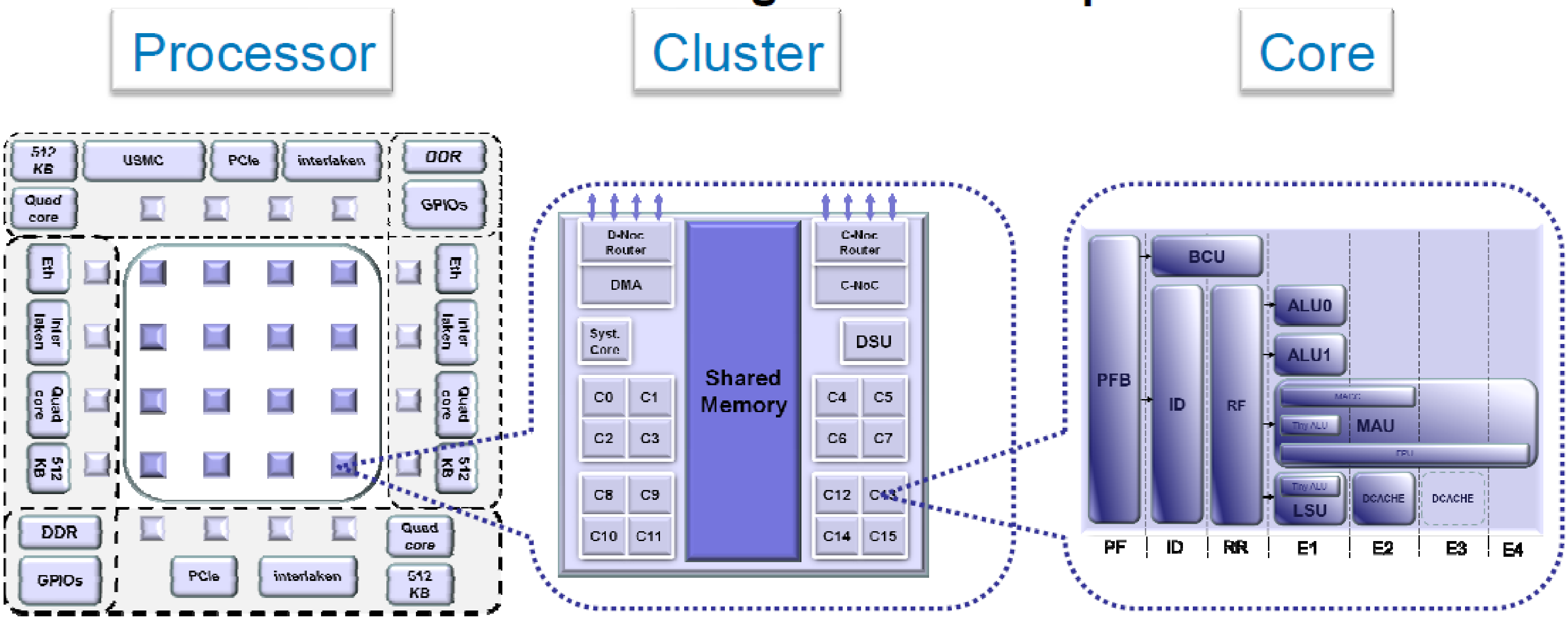
- Geometric constraints
- Differential Inclusion constraints
- Run on many core computing units as well

For more details about Ibex, please refer to École des Mines de Nantes and ENSTA Bretagne

Many Core Computing Architecture

MPPA-256

2.8 Billions transistors, 256 user cores onto a single silicon chip



- NoC to connect clusters
- 40MB of total Memory
- 4x Quad core SMP

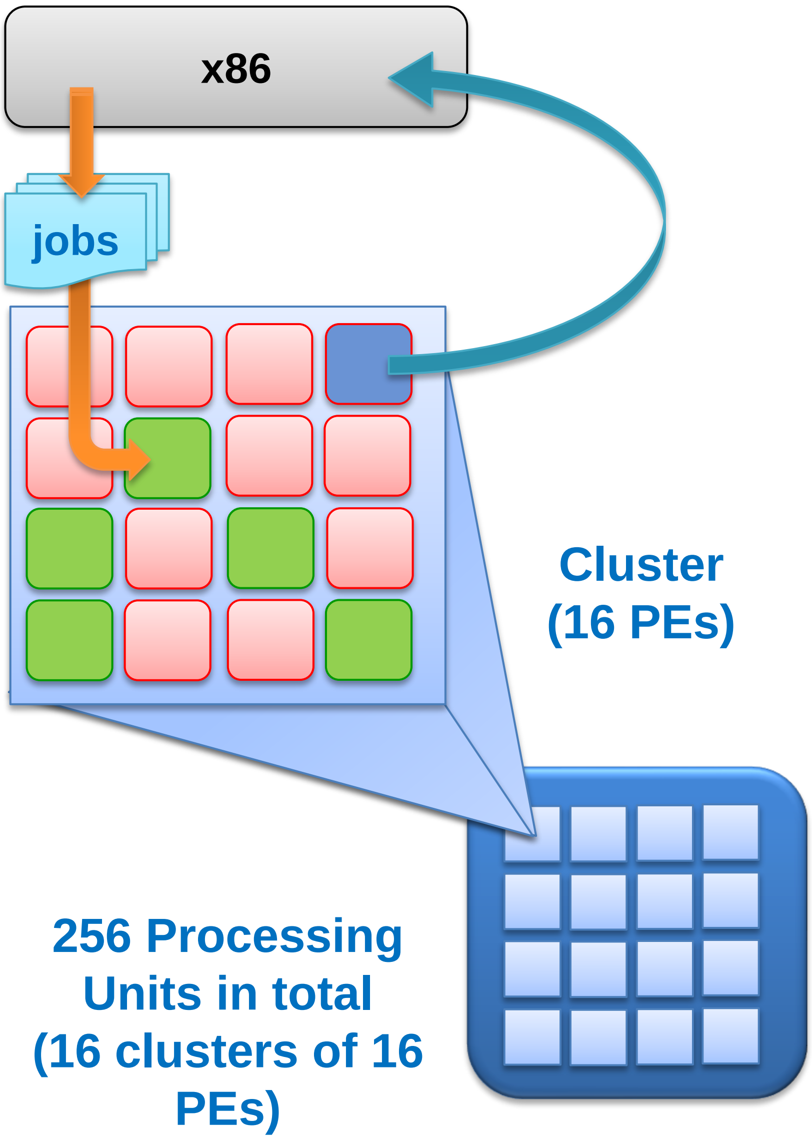
- 28nm Silicon proven
- 16+1 cores by cluster

- High power efficiency
- FPU : 32bits / 64 bits IEEE 754

Powerful and low consumption computing units allowing to (re)compute viability kernels / reachability in real time

Streamer

- Streamer
  - The host processor (x86, IO cluster) sends jobs into a waiting queue
  - Jobs run automatically on the available ressources (Processing Units, PE)
  - Results are sent back to Host using callbacks mechanisms



The Car On the Hill Benchmark Problem

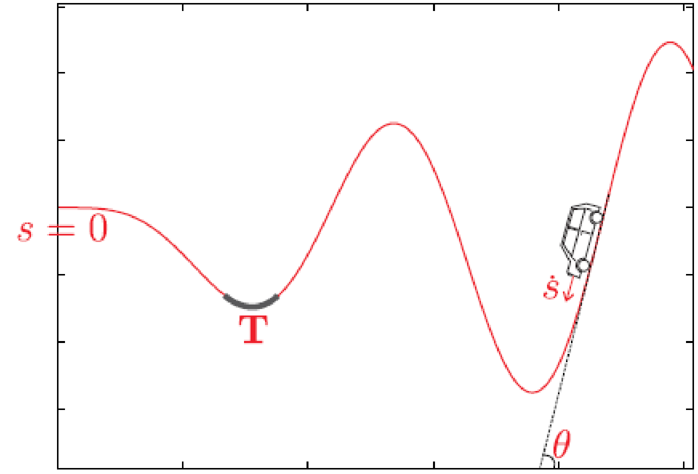
The landscape is represented by the parametric function

$$g : s \rightarrow \frac{\frac{-1.1}{1.2}cos(1.2s) + \frac{1.2}{1.1}cos(1.1s)}{2}$$

State vector: $\mathbf{x} = \begin{pmatrix} s \\ \dot{s} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

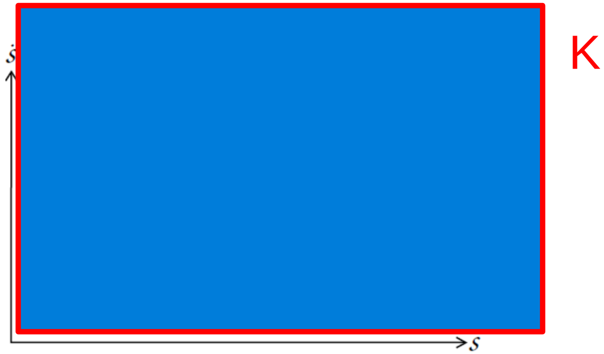
Evolution function:

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = -9.81 sin(\frac{dg}{dx_1}(x_1)) - 0.7x_2 + u \end{cases}$$



$u \in [-7, 7]$

The car must stay on the landscape, i.e $s \in [-1, 13]$, $\dot{s} \in [-7, 7]$



Interval Based Viability Kernel Algorithm
The Bisection Based Approach

Consider the differential inclusion F
The set of constraints K
The target set C

$$x_{n+1} \in F(x_n)$$

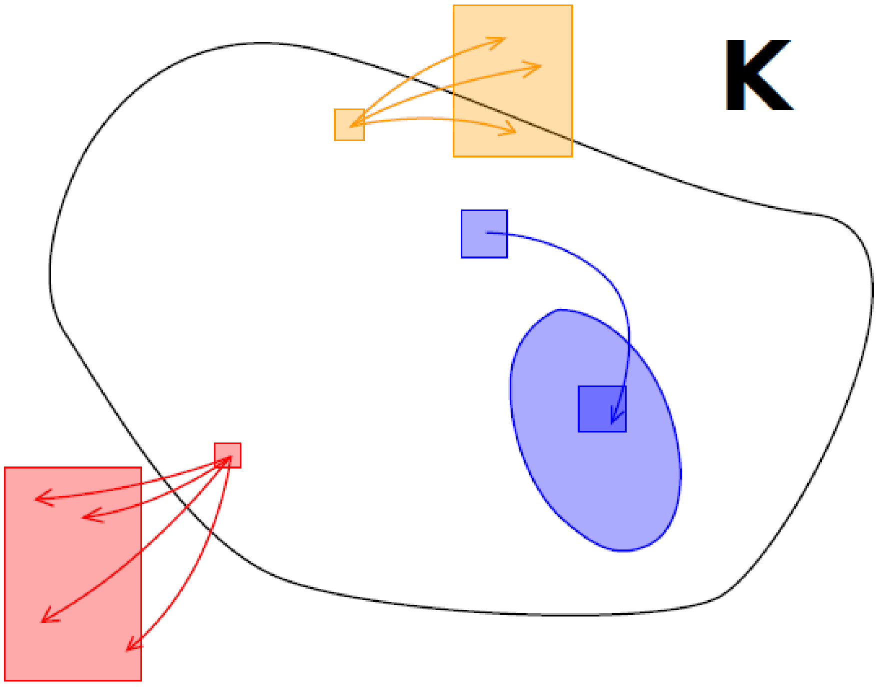
$$(C \subset K)$$

$$V = Viab_F(K, C) = Capt_F(K, C)$$

(which is always true for non stationary systems)

Then, the viability algorithm starting from K and decreasing to an over approximation of V is given by the following algorithm :

$$\begin{cases} K_0 = K \\ K_n = (K_{n-1} \cap F^{-1}\,(K_{n-1})) \cup C \end{cases}$$



Drawing performed by Dominique Monnet

Interval Based Viability Kernel Algorithm
The Contractor Approach

Consider the following Differential Inclusion constraint :

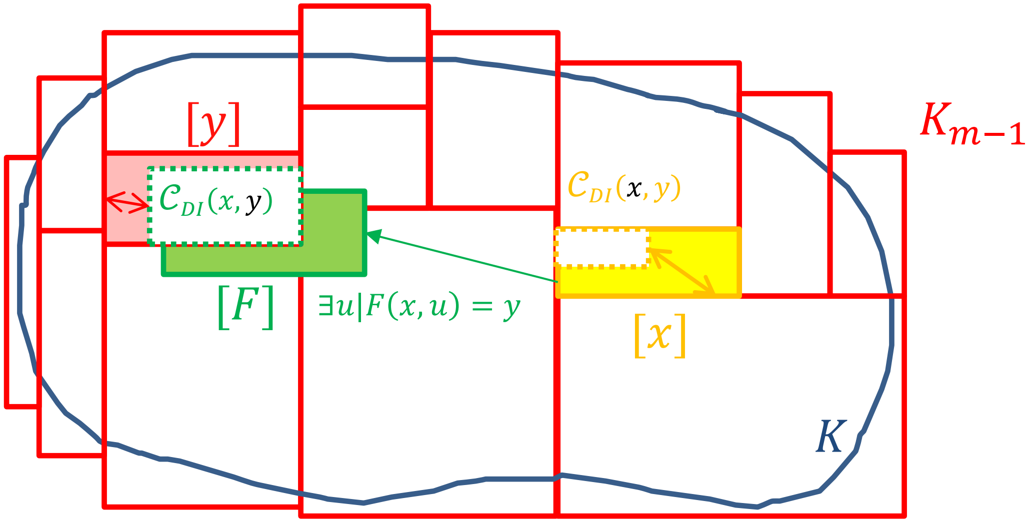
$$\begin{cases} x = x_n \\ y = x_{n+1} \end{cases}$$

$$DI\ constraint\ (x, y) : \exists u \in [u] \mid F(x, u) = y$$

And the associated contractor : $\mathcal{C}_{DI}(x, y)$

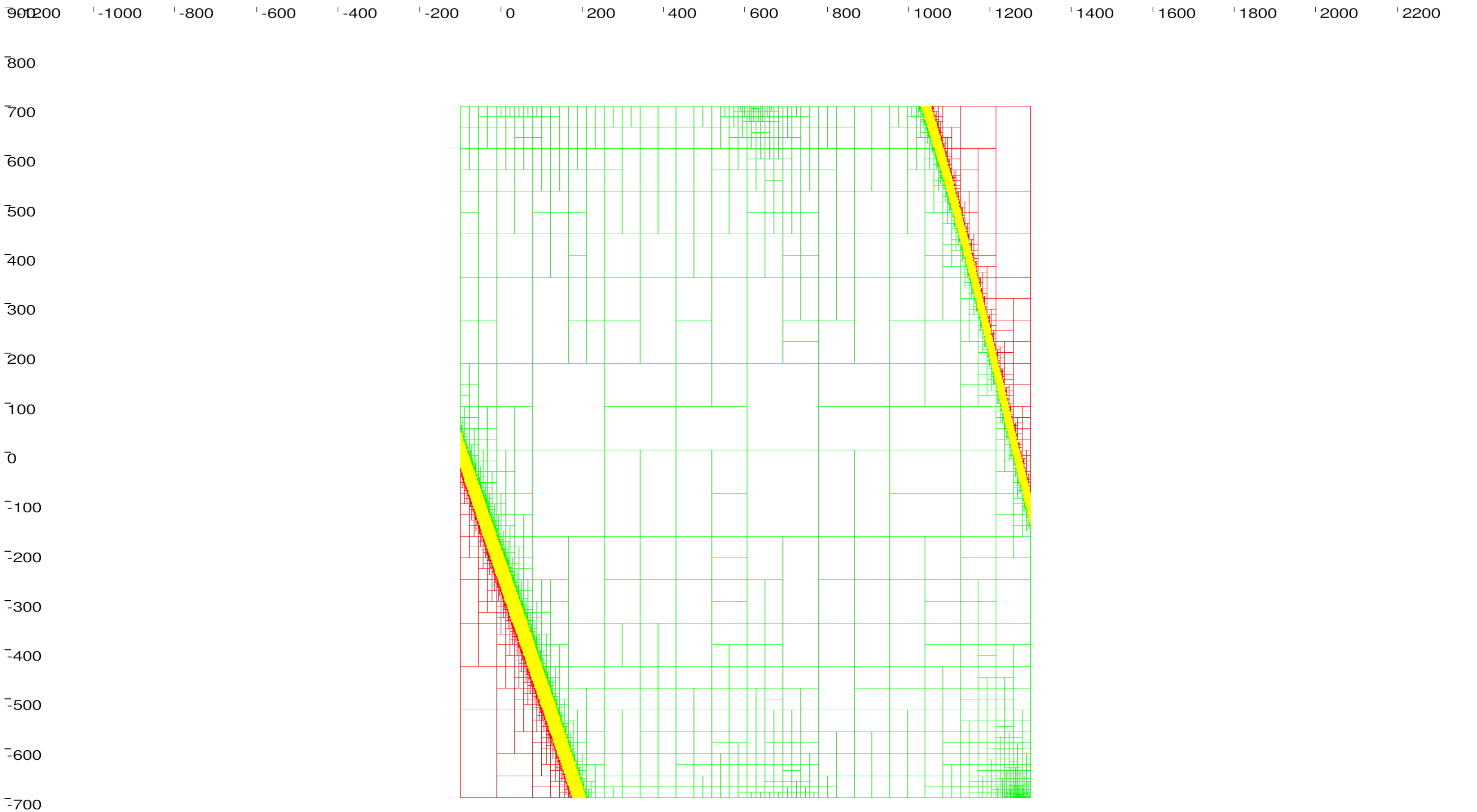
Then, the viability kernel K_m can be computed in an iterative manner as follow:

$$K_m^{Comp} = \left(\bigcup_{[x] \subset K_{m-1}/\mathcal{C}_n} \bigcap_{[y] \subset K_{m-1}} ([x] - Proj_x(\mathcal{C}_{DI}(x, y))) \right) \bigcup K_{m-1}^{Comp}$$



$$\varepsilon = 0.01\ (m, box\ size)$$
$$dt = 0.01(\text{sec}, \text{time step})$$
$$nb\ pas\ traj = 50$$
$$Euler$$

Numerical Results
Viability Kernel
Bisection - K1 - One Iteration Only



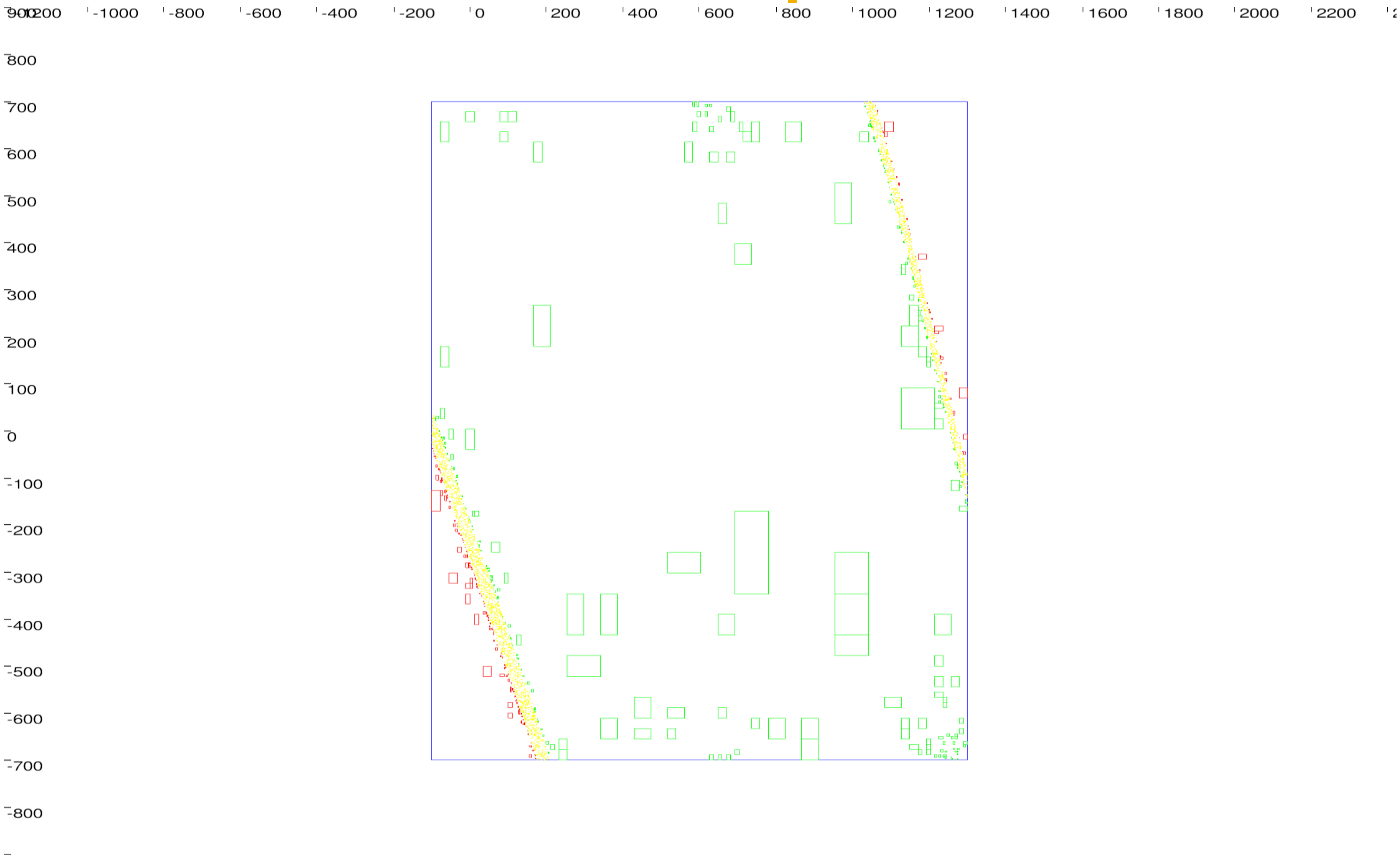
⇒ very fast to compute, nice results, but viability kernel over estimated

Due to long trajectory integration process, we can rely on $Viab_F^{Comp}(K, C)$ only !!!

Numerical Results
Parallel Computing
Bisection - K1 - One Iteration Only

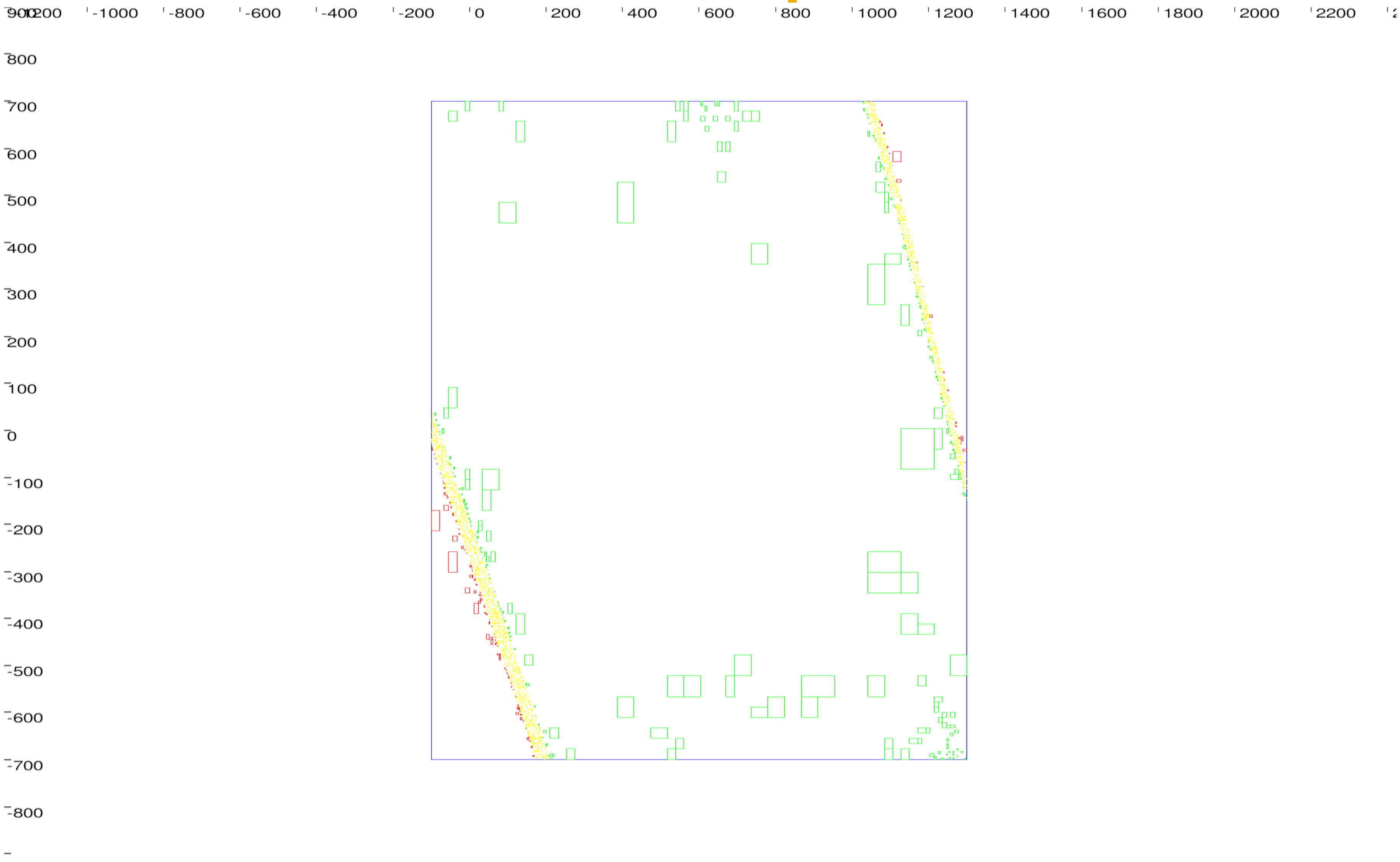
```
File  Edit  View  Search  Terminal  Help
Cluster 10 PE 3
Cluster 0 PE 1
Cluster 11 PE 1
Cluster 10 PE 0
Cluster 3 PE 1
Cluster 2 PE 3
Cluster 8 PE 2
Cluster 9 PE 1
Cluster 0 PE 2
Cluster 12 PE 2
Cluster 4 PE 2
Cluster 1 PE 1
Cluster 11 PE 2
Cluster 3 PE 2
Cluster 9 PE 2
Cluster 1 PE 2
Cluster 10 PE 1
Cluster 2 PE 1
Cluster 10 PE 2
Cluster 2 PE 2
All requests have been processed
****************************************************************
Total time for processing 101029 requests : 58729.8 ms
Average time per request : 0.581316 ms
****************************************************************
[accesscore@mppa-dev007 Streamer]$
```

Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 0

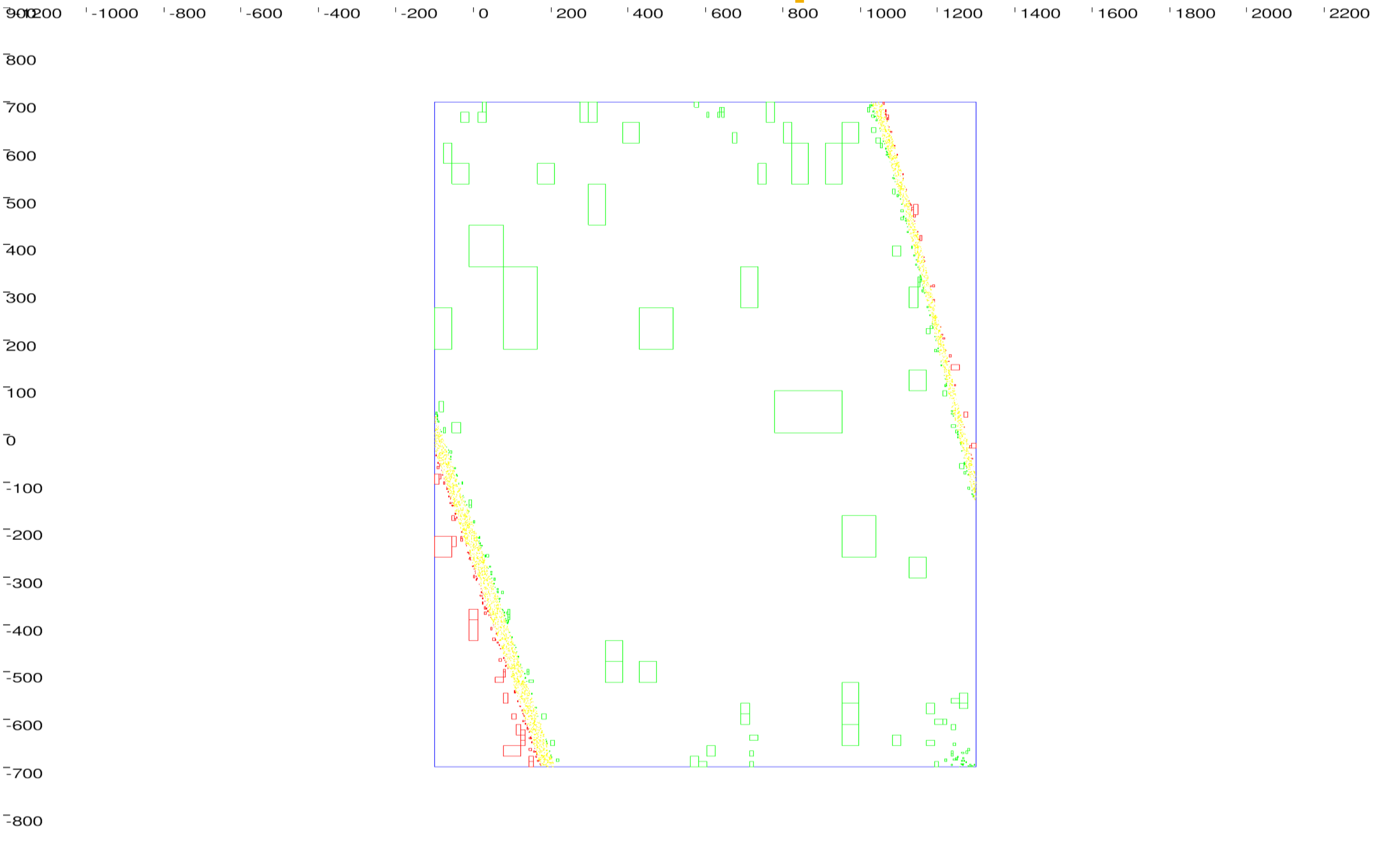


Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 1

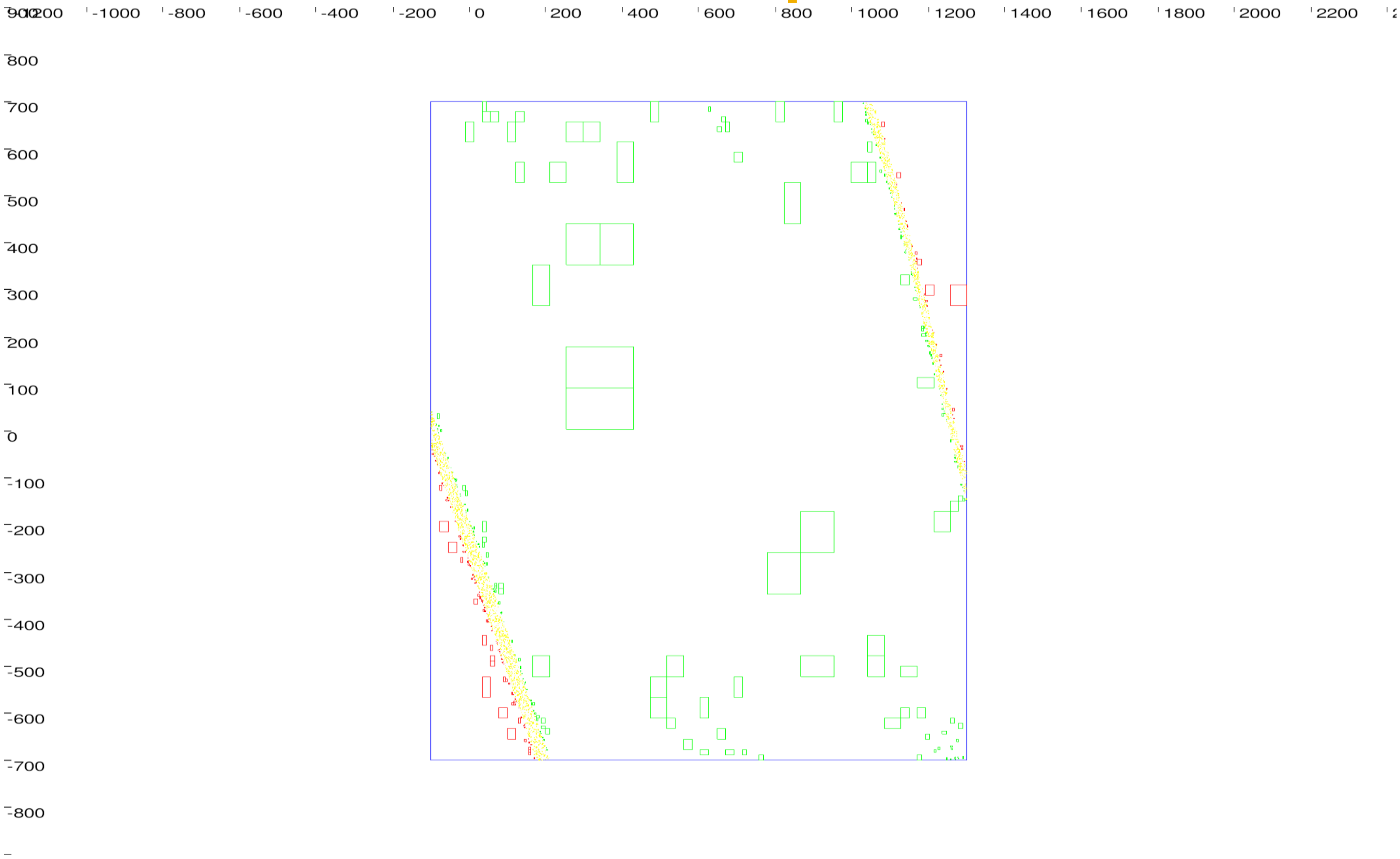
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 2



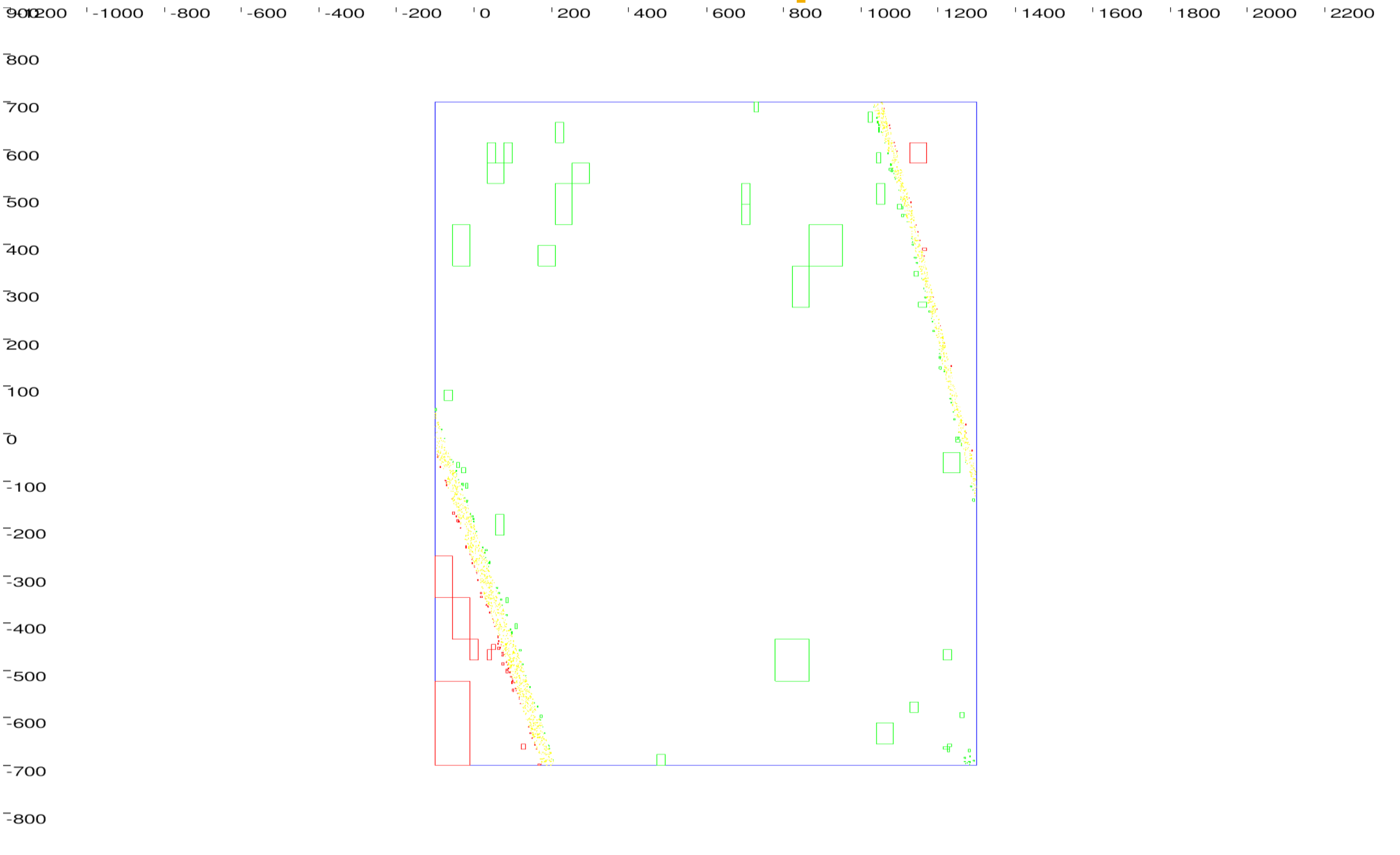
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 3



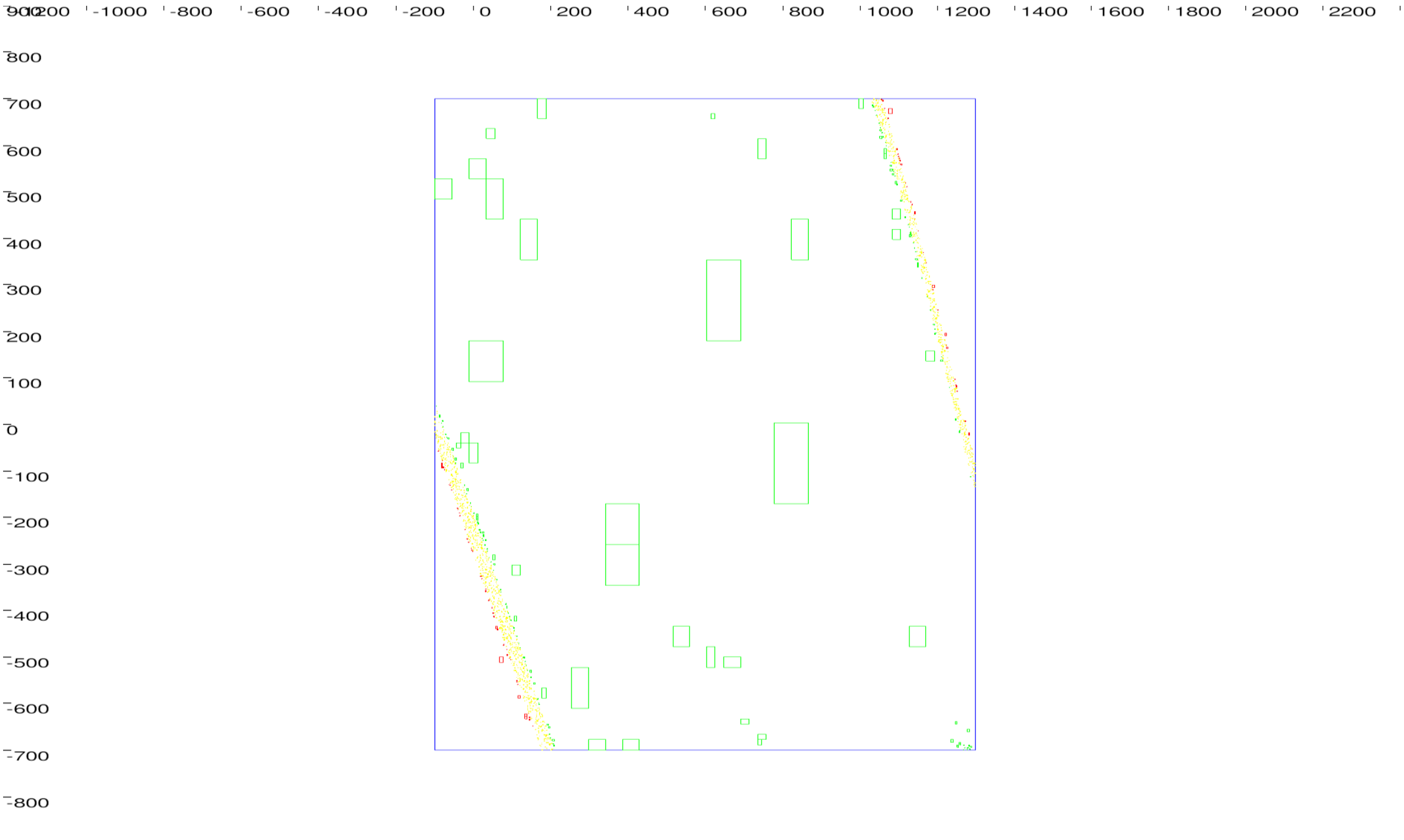
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 4



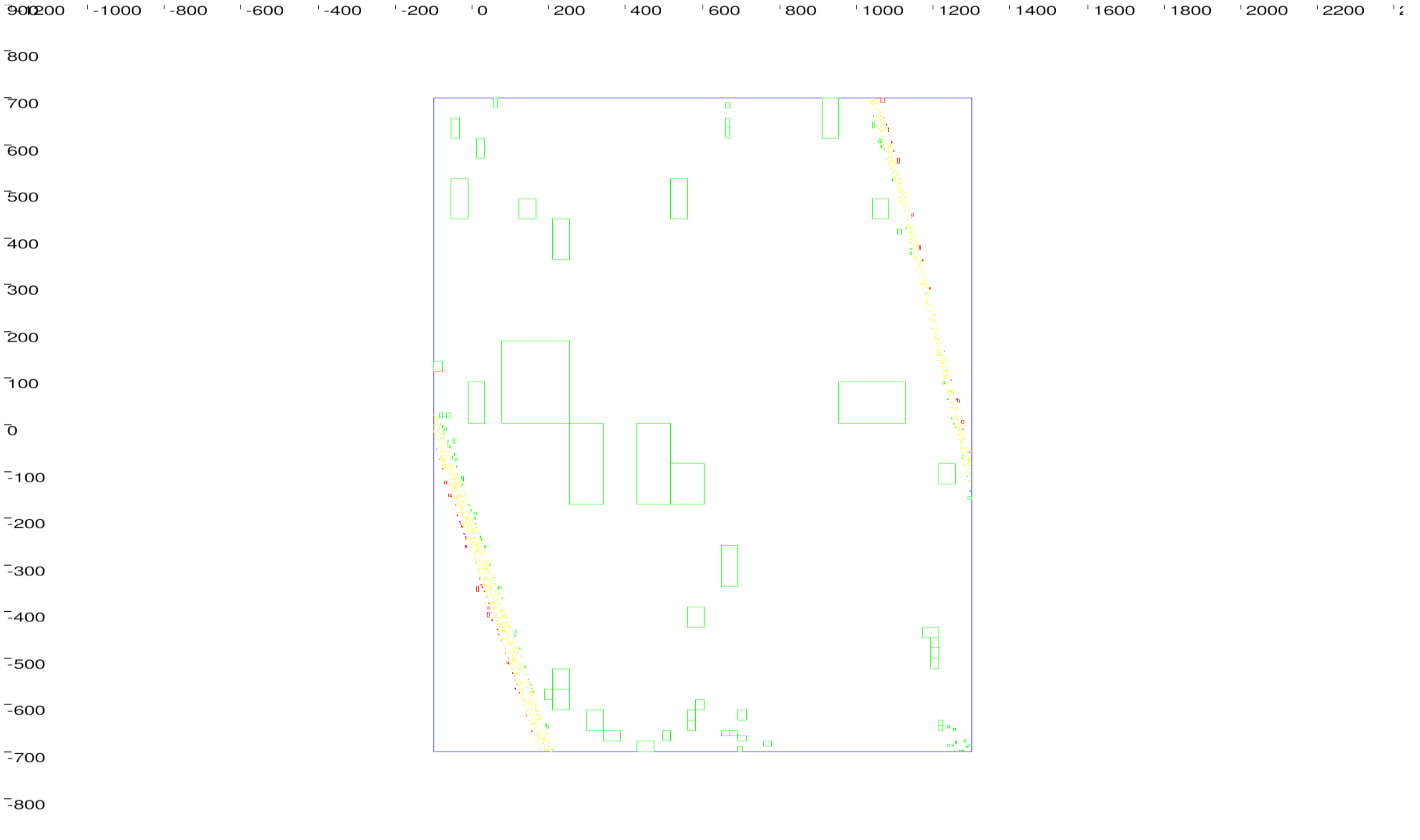
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 5



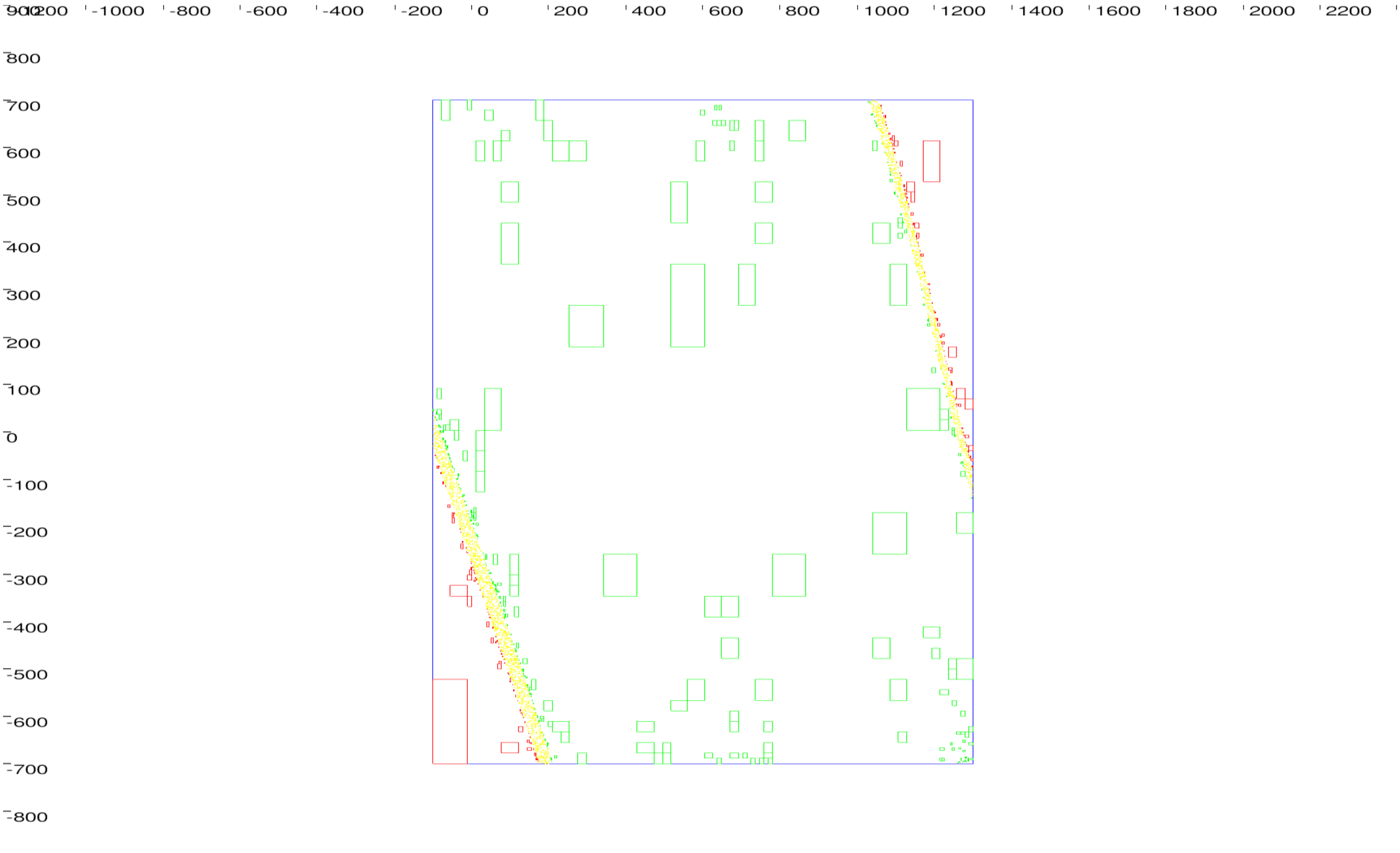
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 6



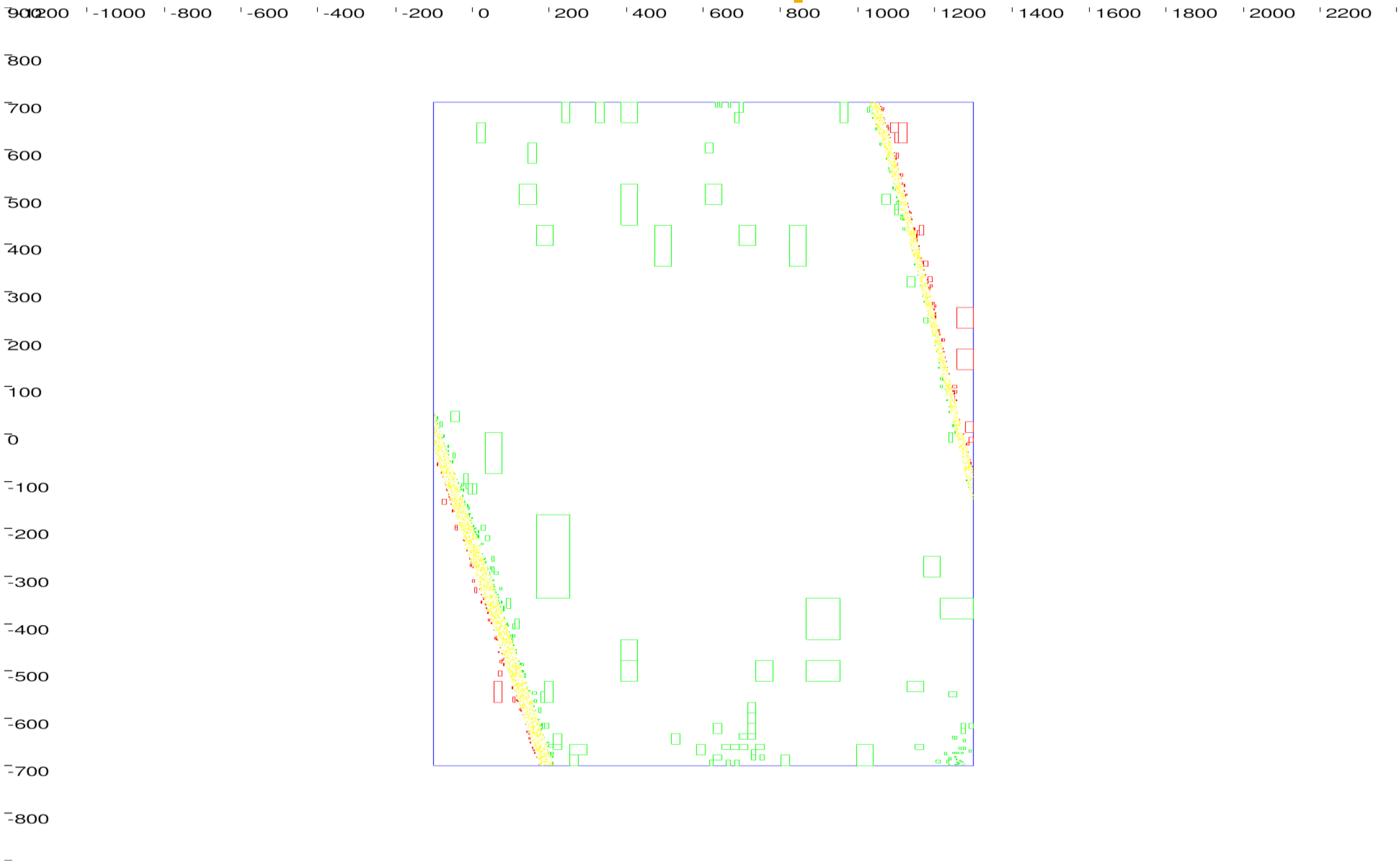
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 7



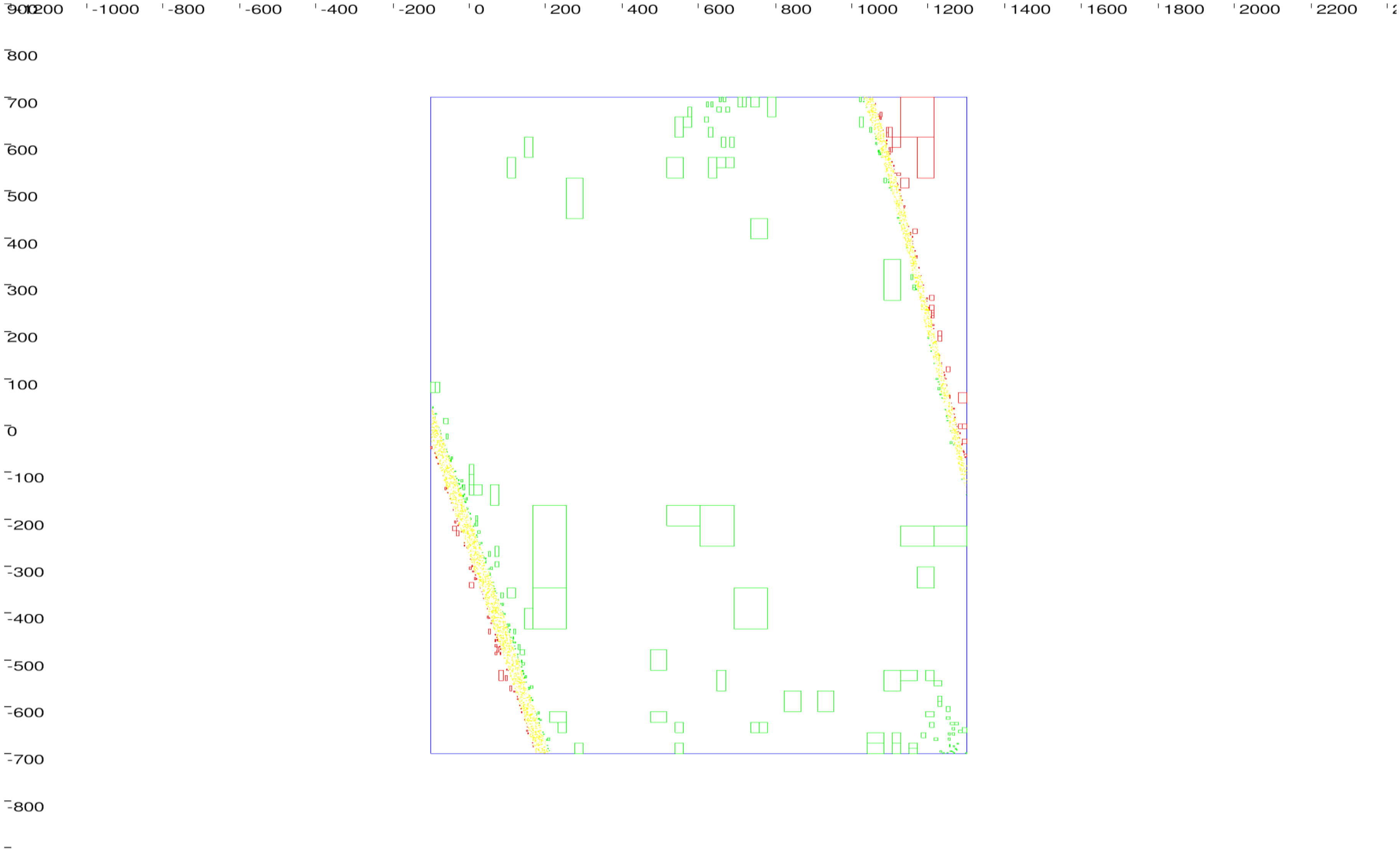
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 8



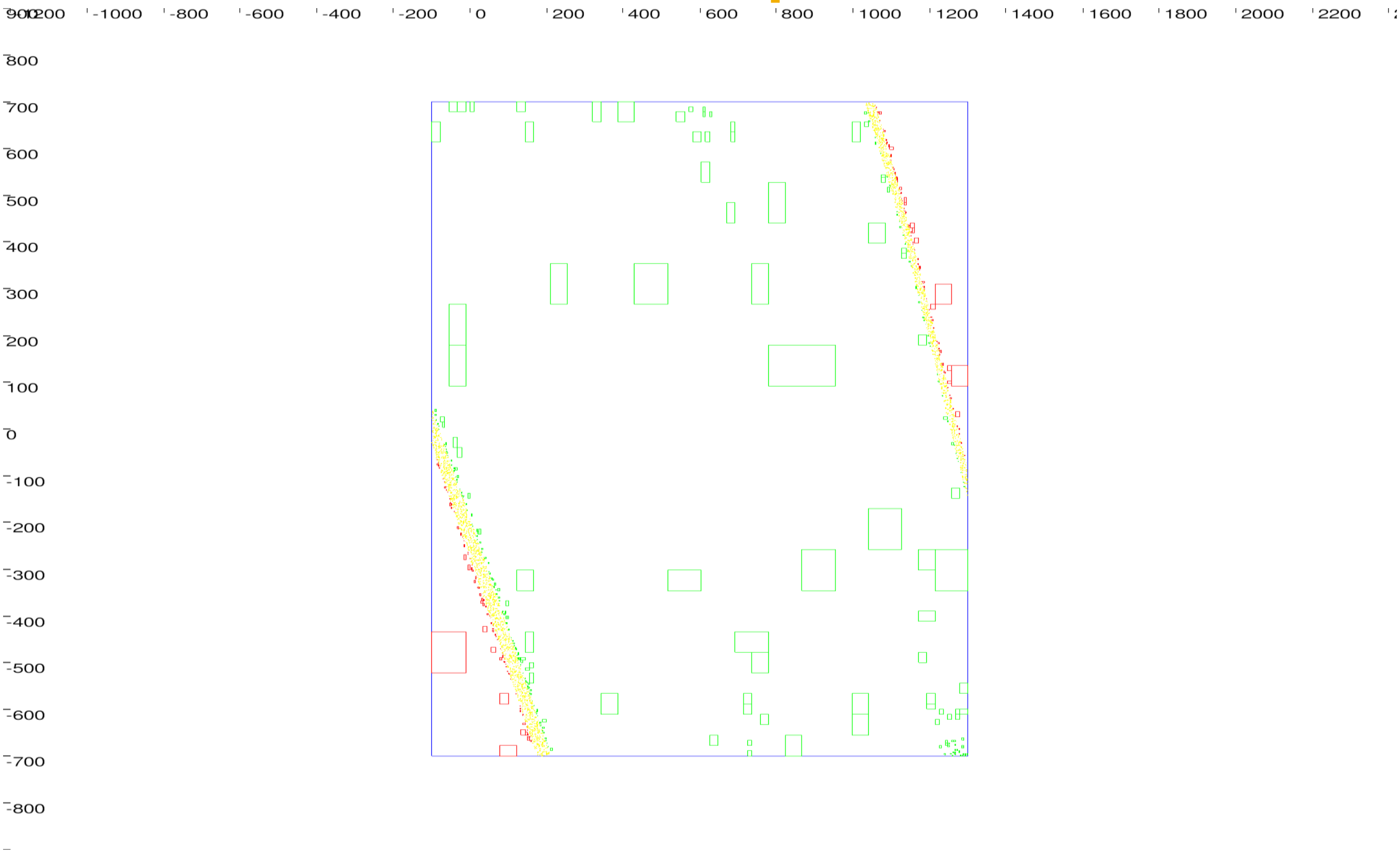
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 9



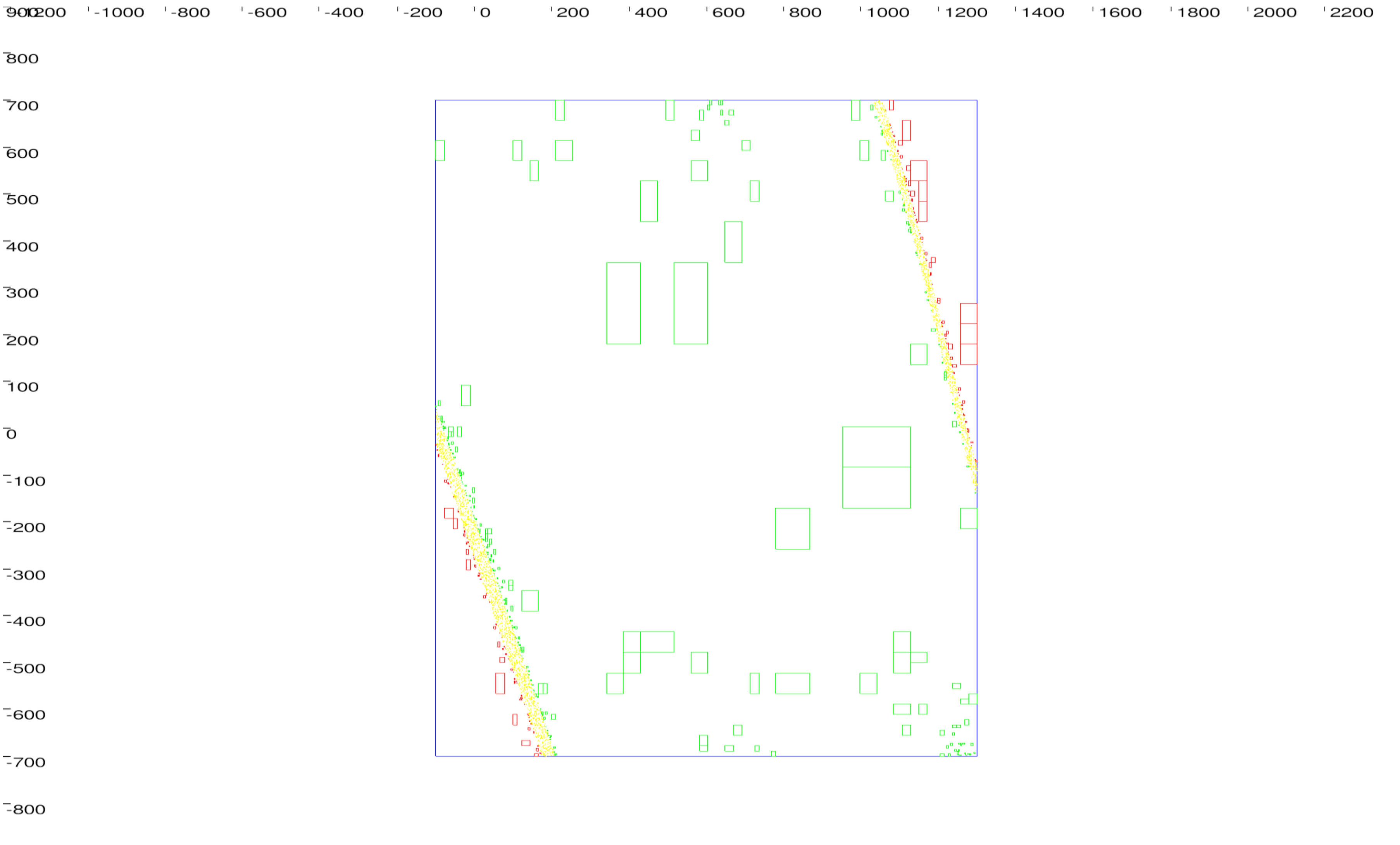
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 10



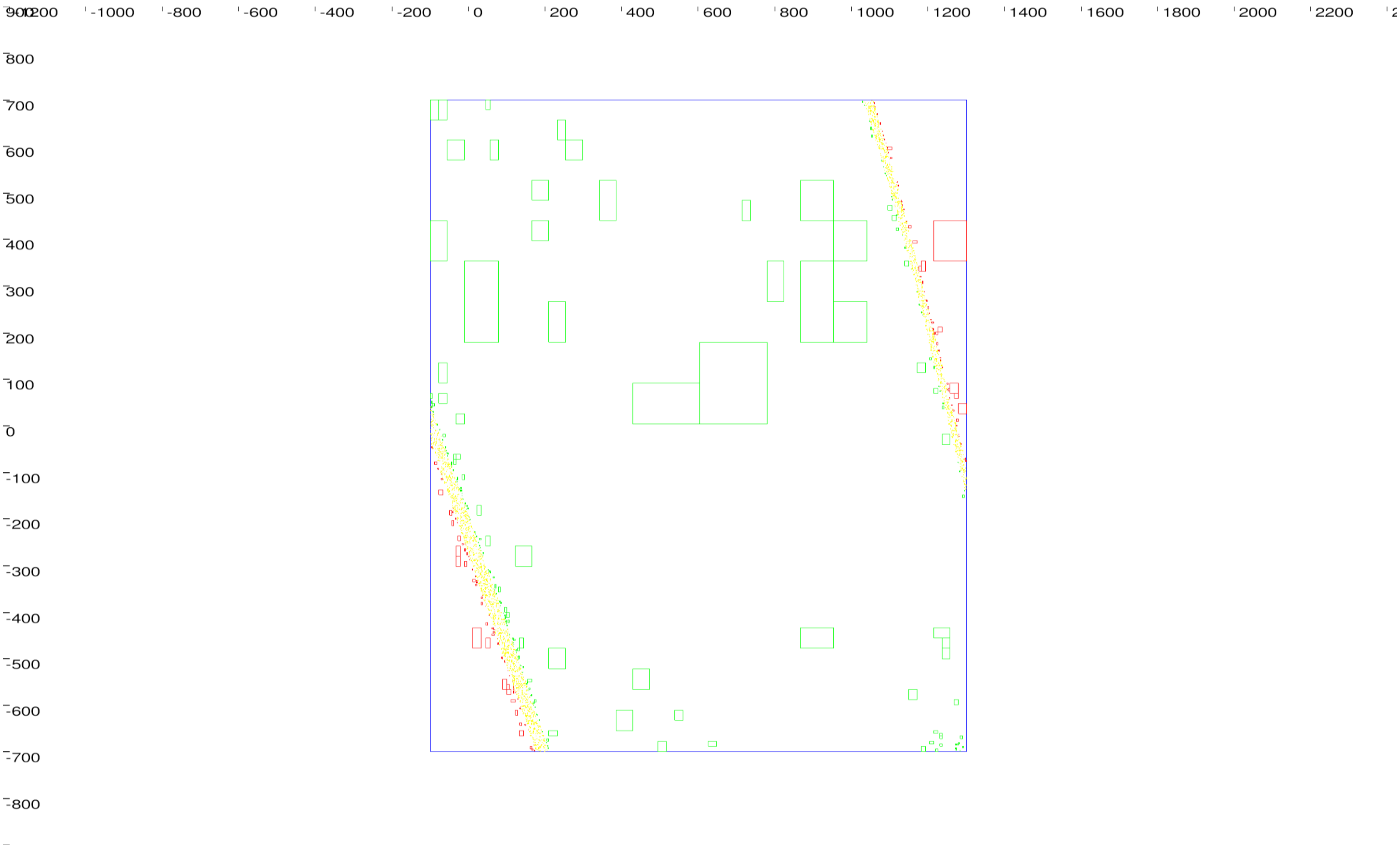
Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 11



Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 12

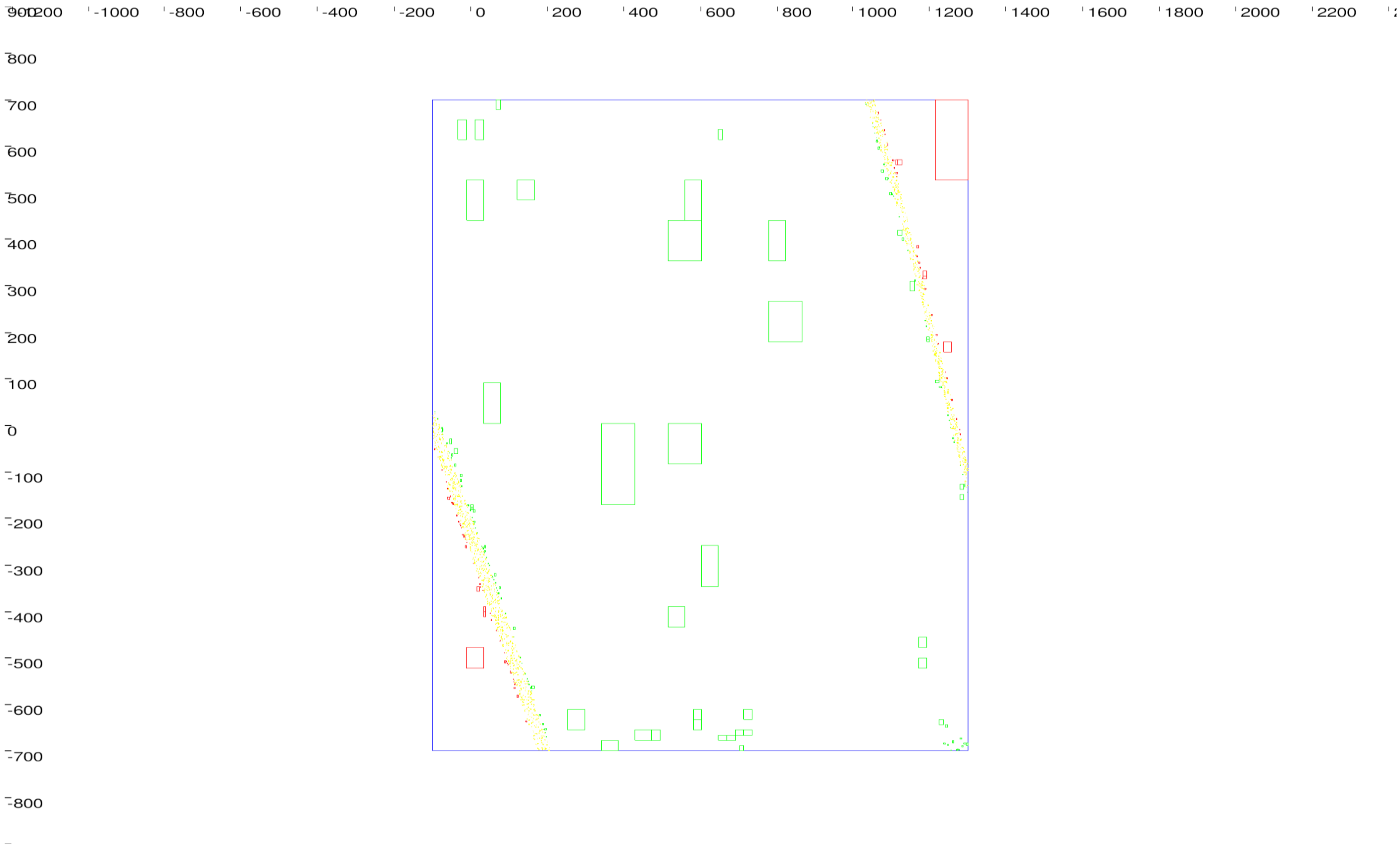


Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 13

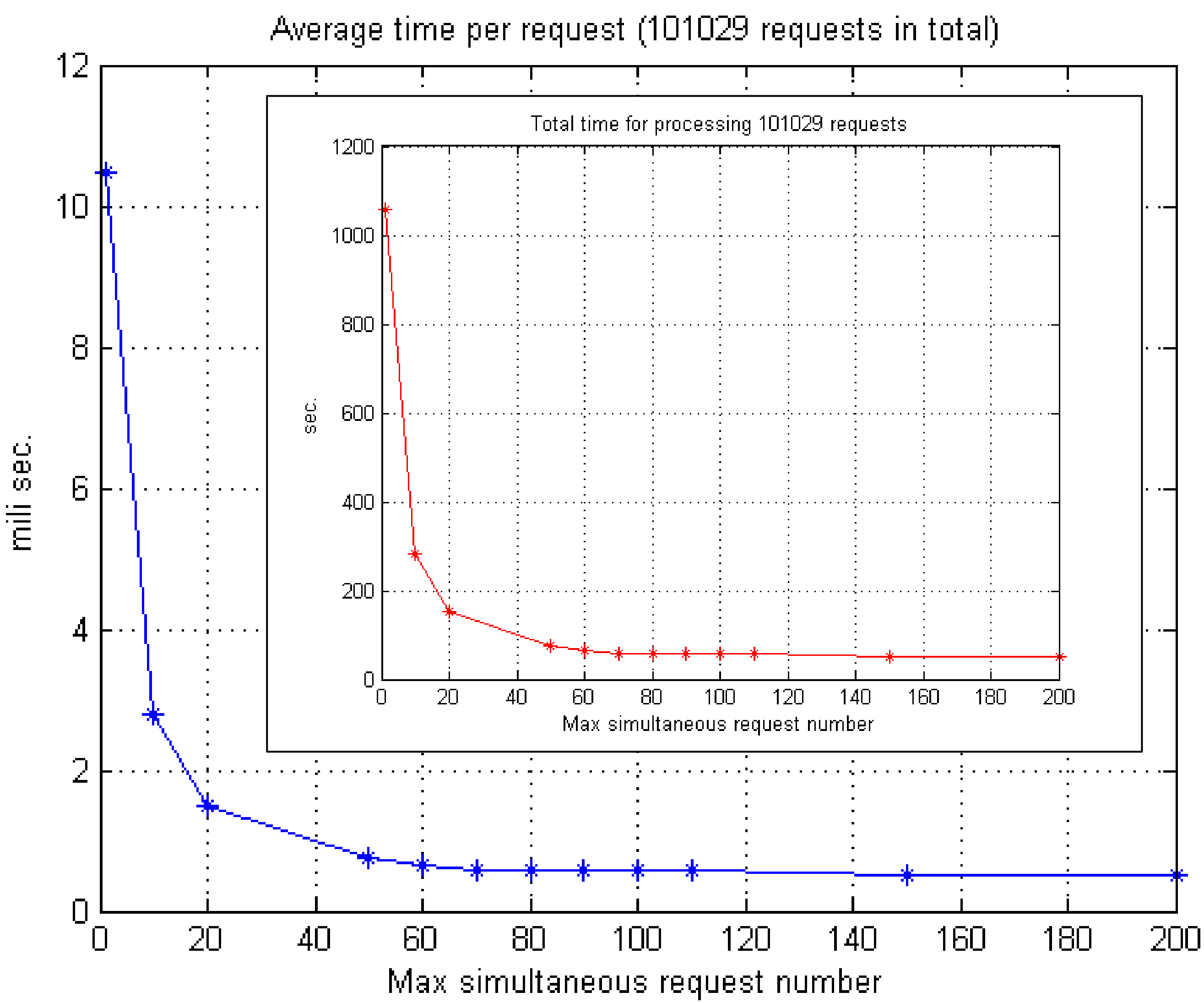


Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 14

Numerical Results
Bisection - K1 - One Iteration Only
Detail per Cluster – Cluster 15

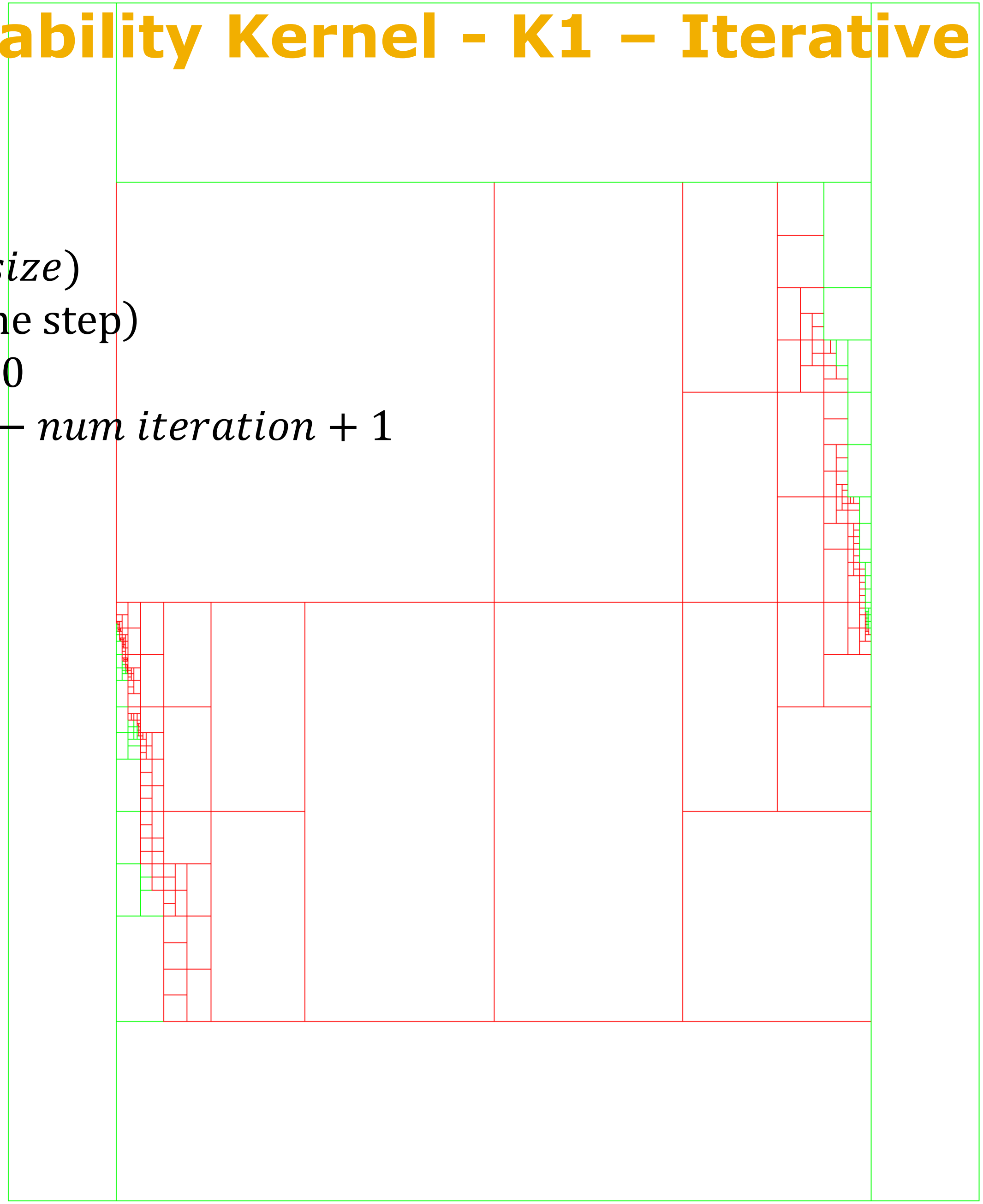


Performance Comparison

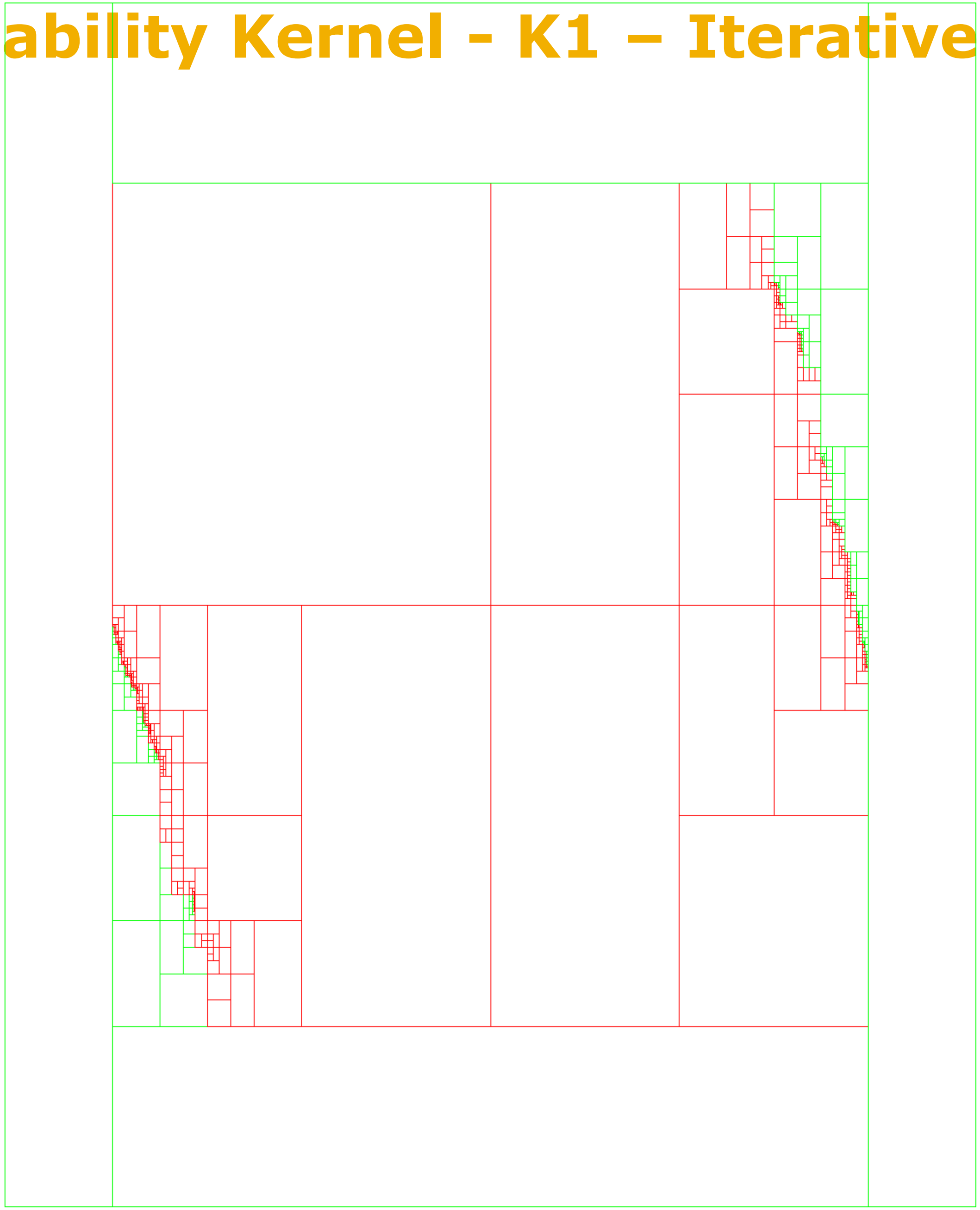


Viability Kernel - K1 – Iterative Computing Iteration 1

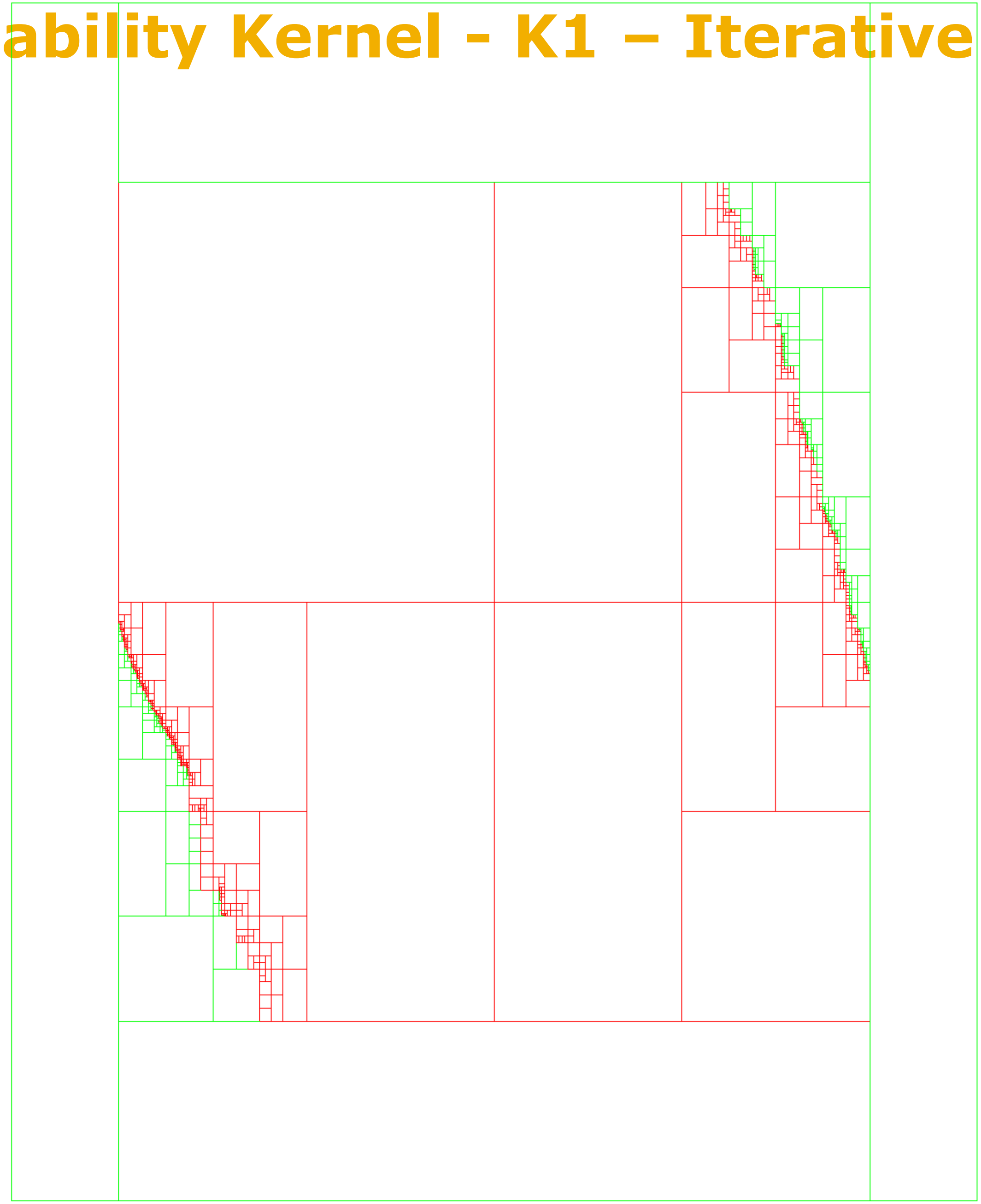
$\varepsilon = 0.05\ (m, box\ size)$
$dt = 0.01\ (\mathrm{sec, time\ step})$
$nb\ iterations = 10$
$nb\ pas\ traj = 10 - num\ iteration + 1$
$Heun$



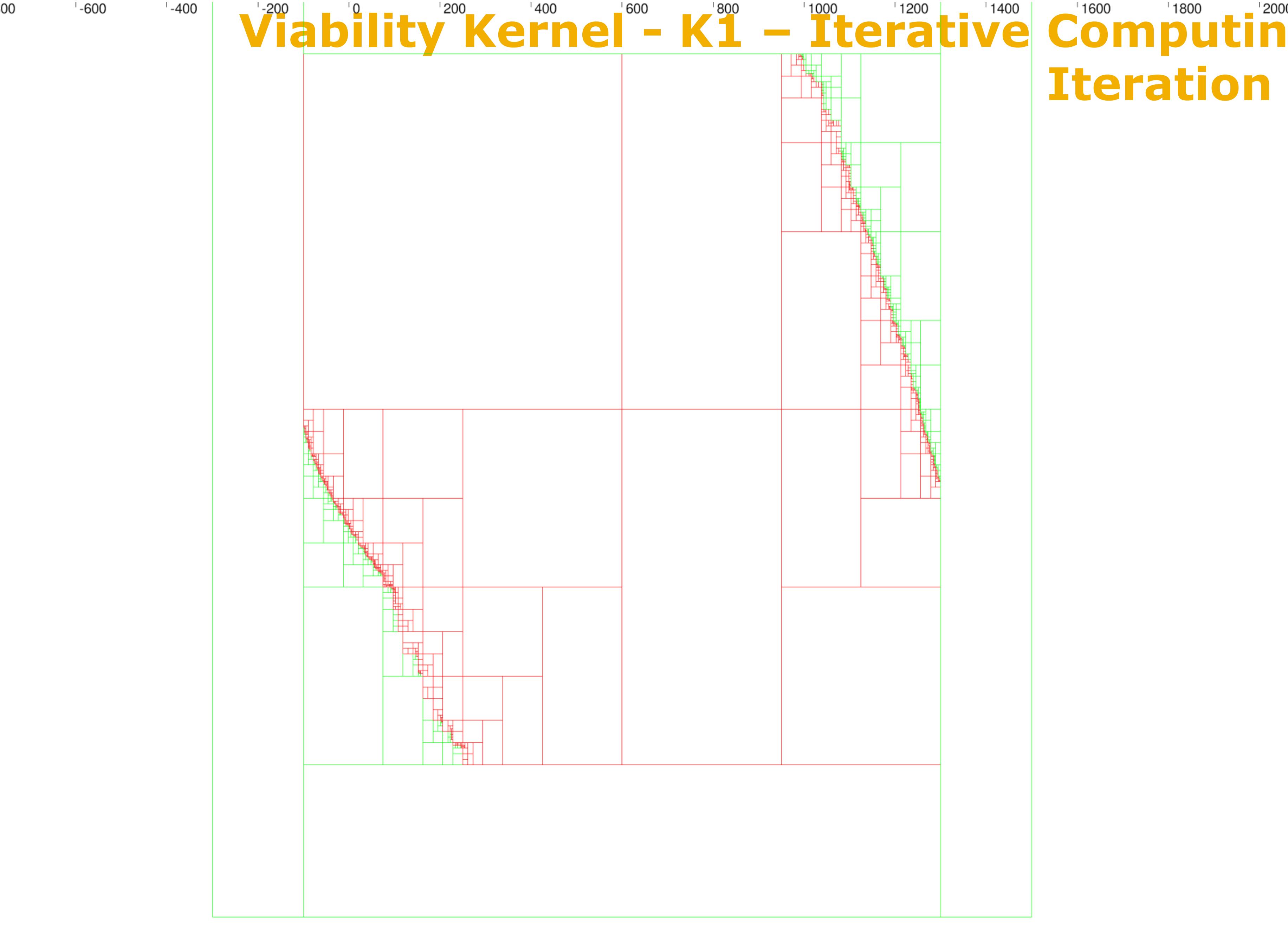
Viability Kernel - K1 – Iterative Computing
Iteration 2



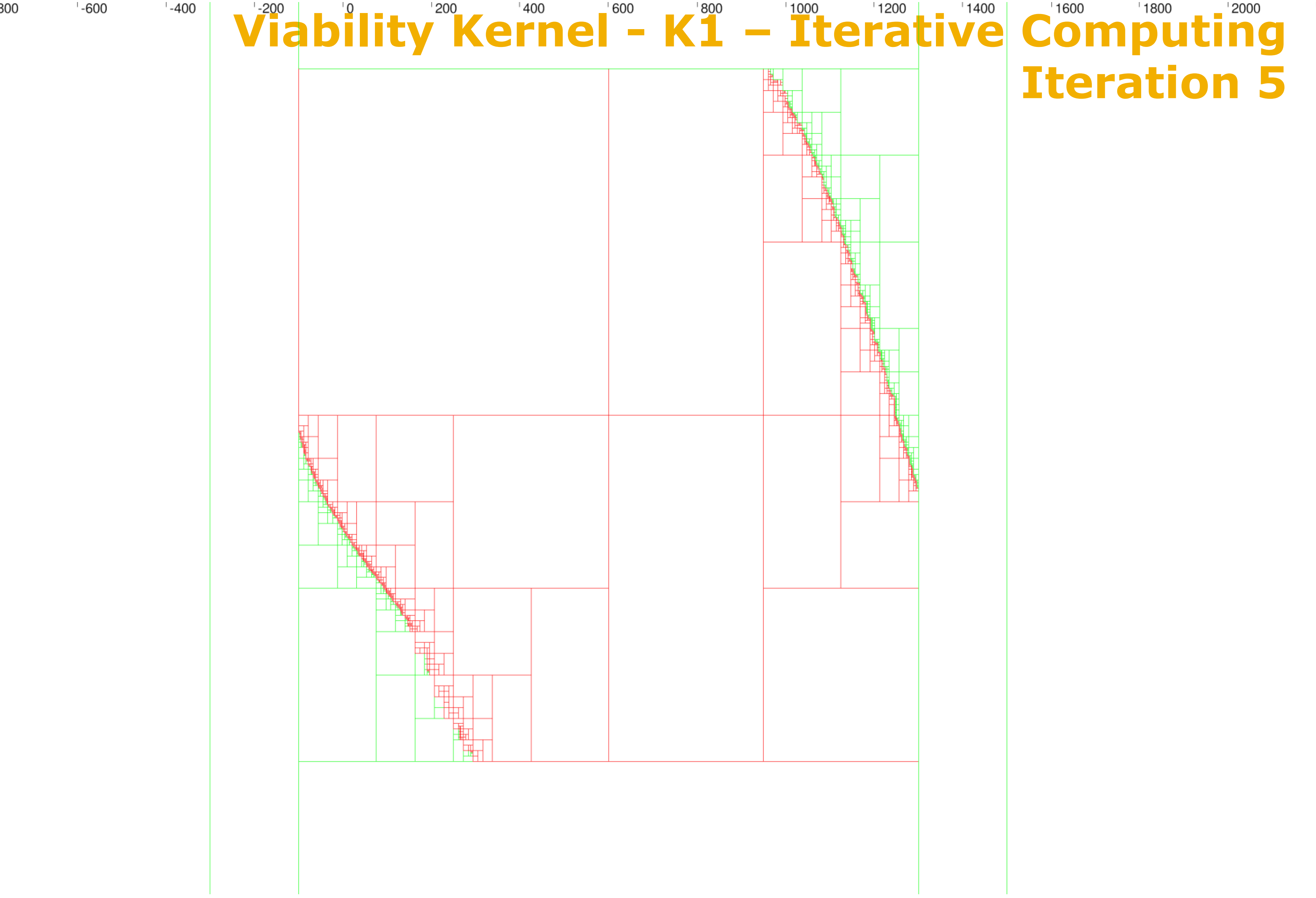
Viability Kernel - K1 – Iterative Computing
Iteration 3



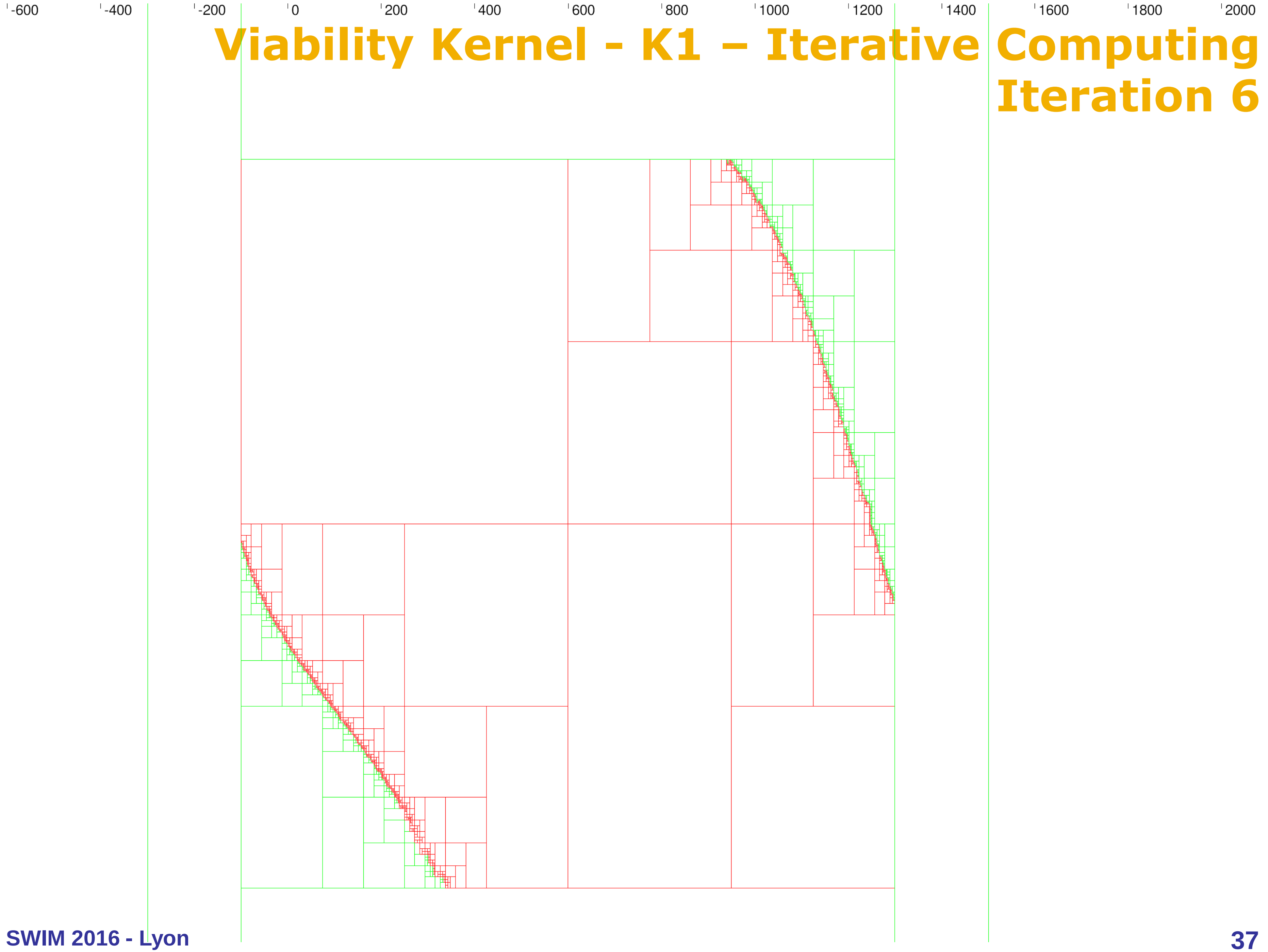
Viability Kernel - K1 – Iterative Computing Iteration 4



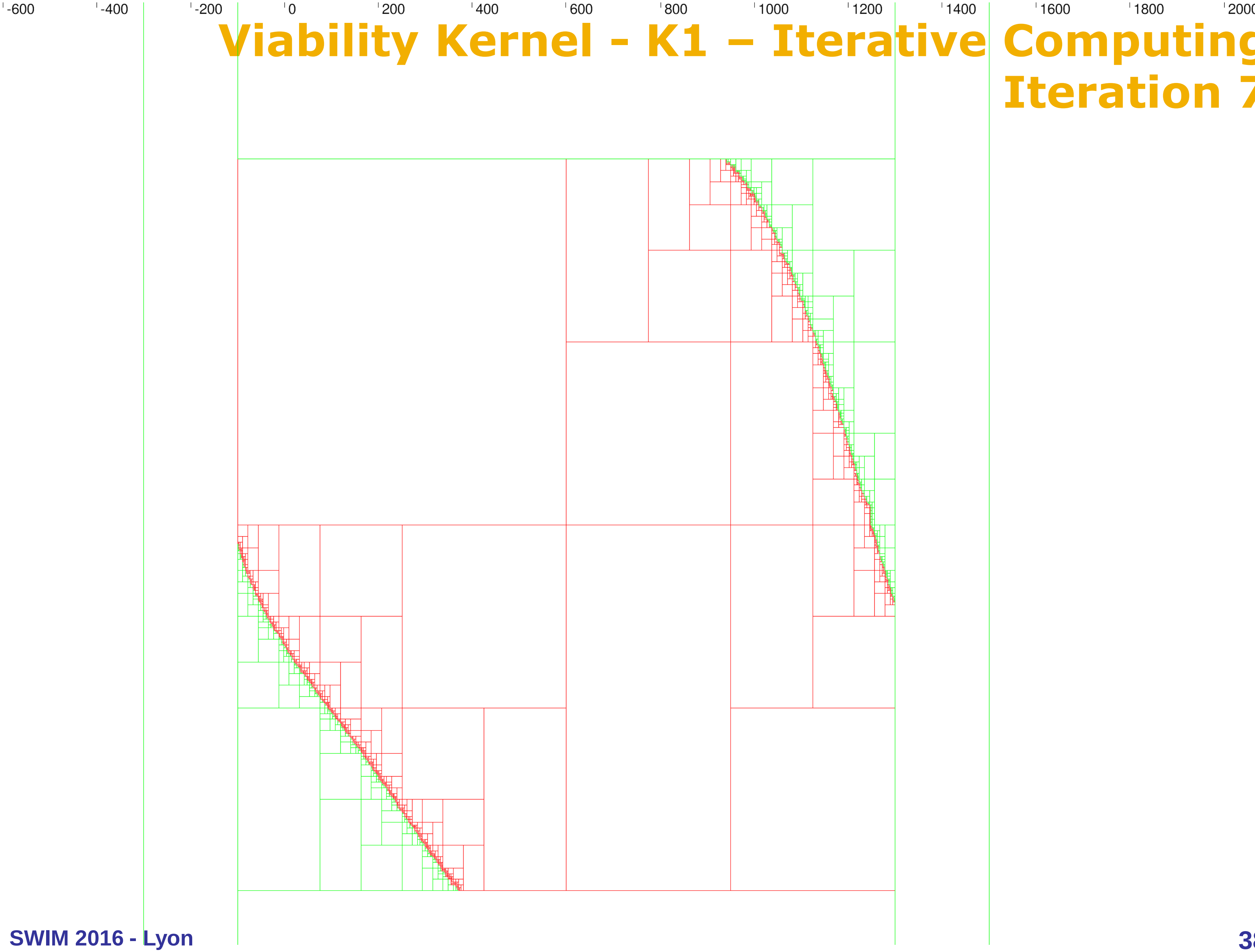
Viability Kernel - K1 – Iterative Computing Iteration 5



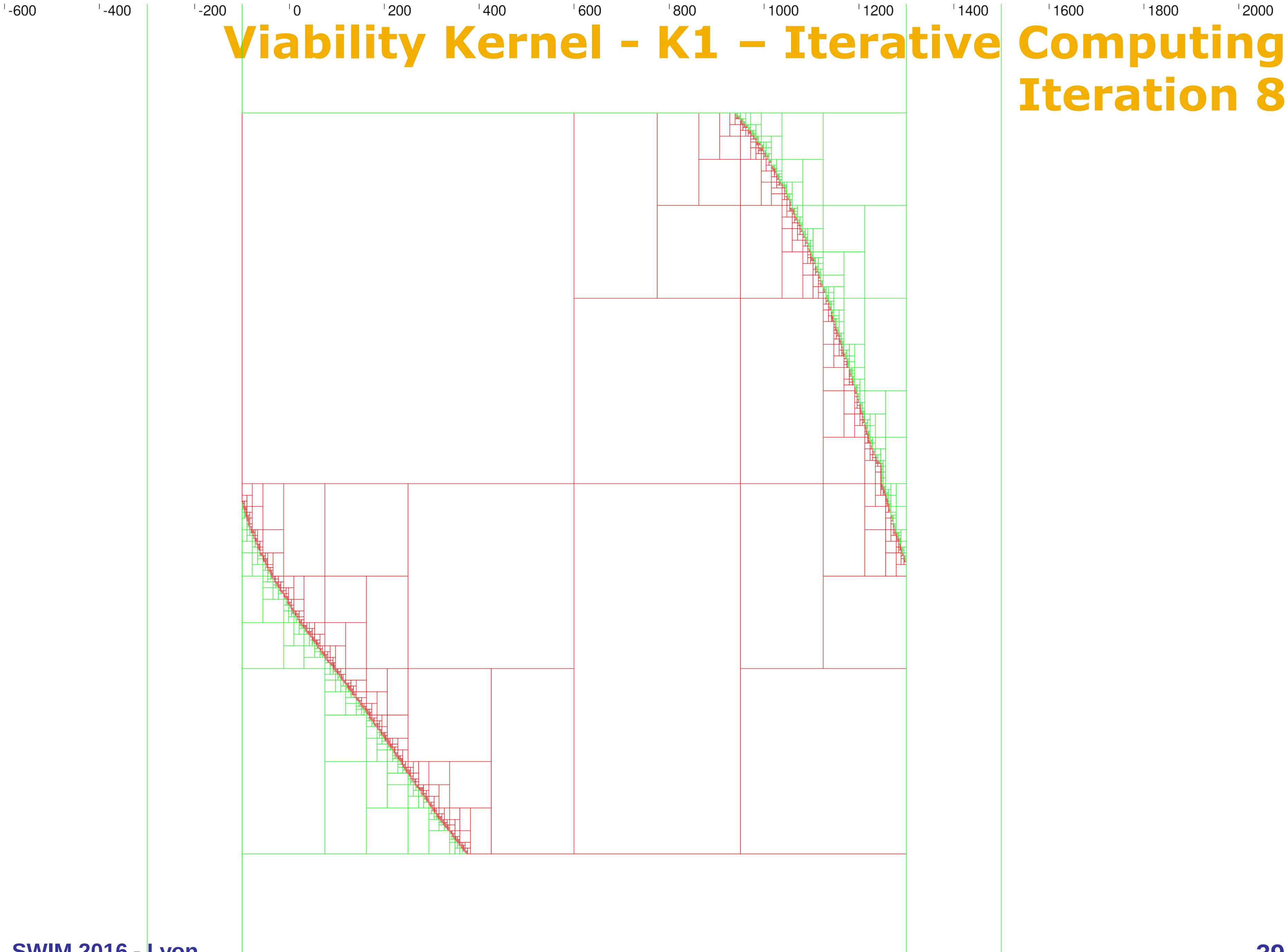
Viability Kernel - K1 – Iterative Computing Iteration 6



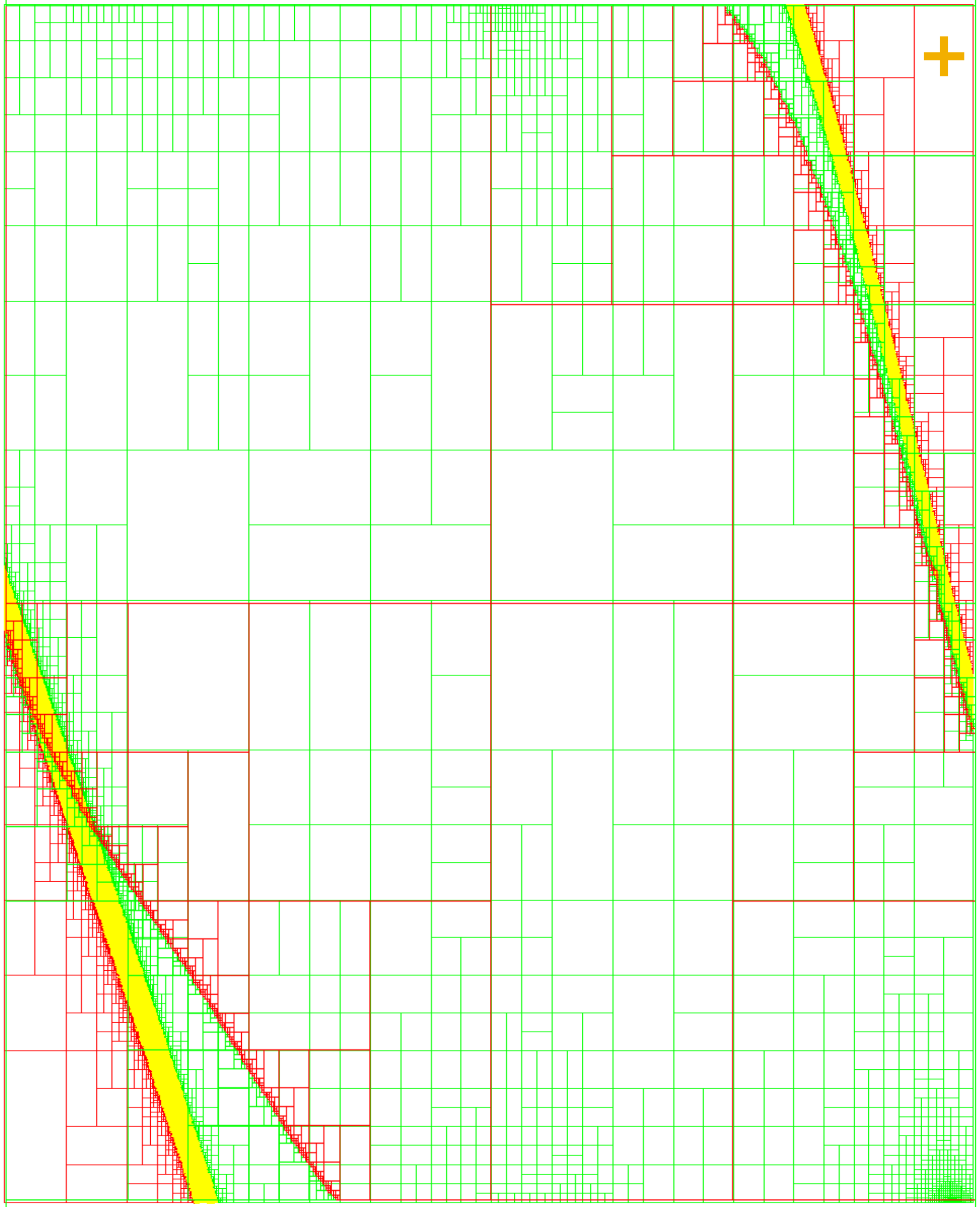
Viability Kernel - K1 – Iterative Computing Iteration 7



Viability Kernel - K1 – Iterative Computing Iteration 8

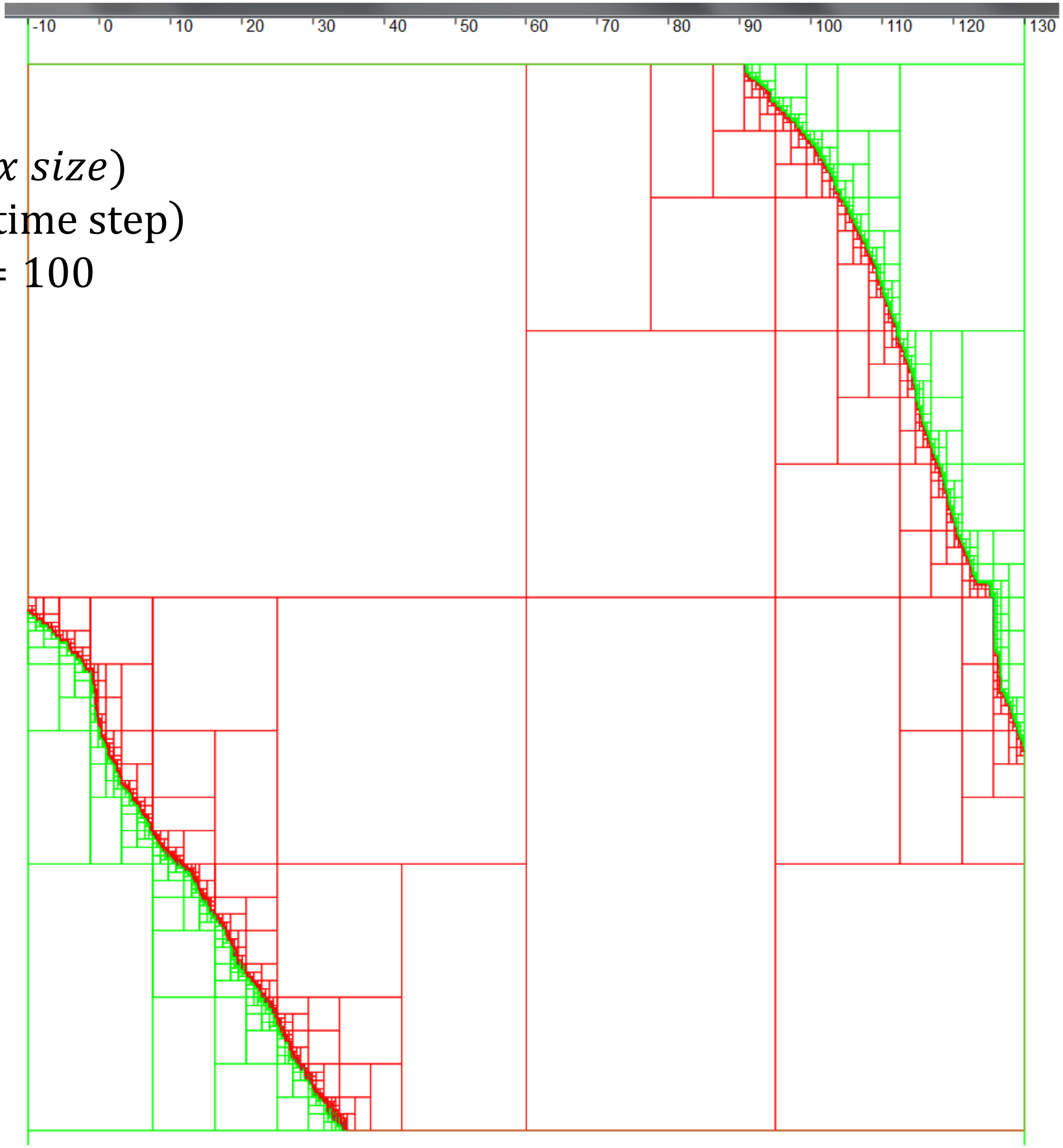


Viability Kernel - K1 – Iterative Computing Iteration 9 + First Results

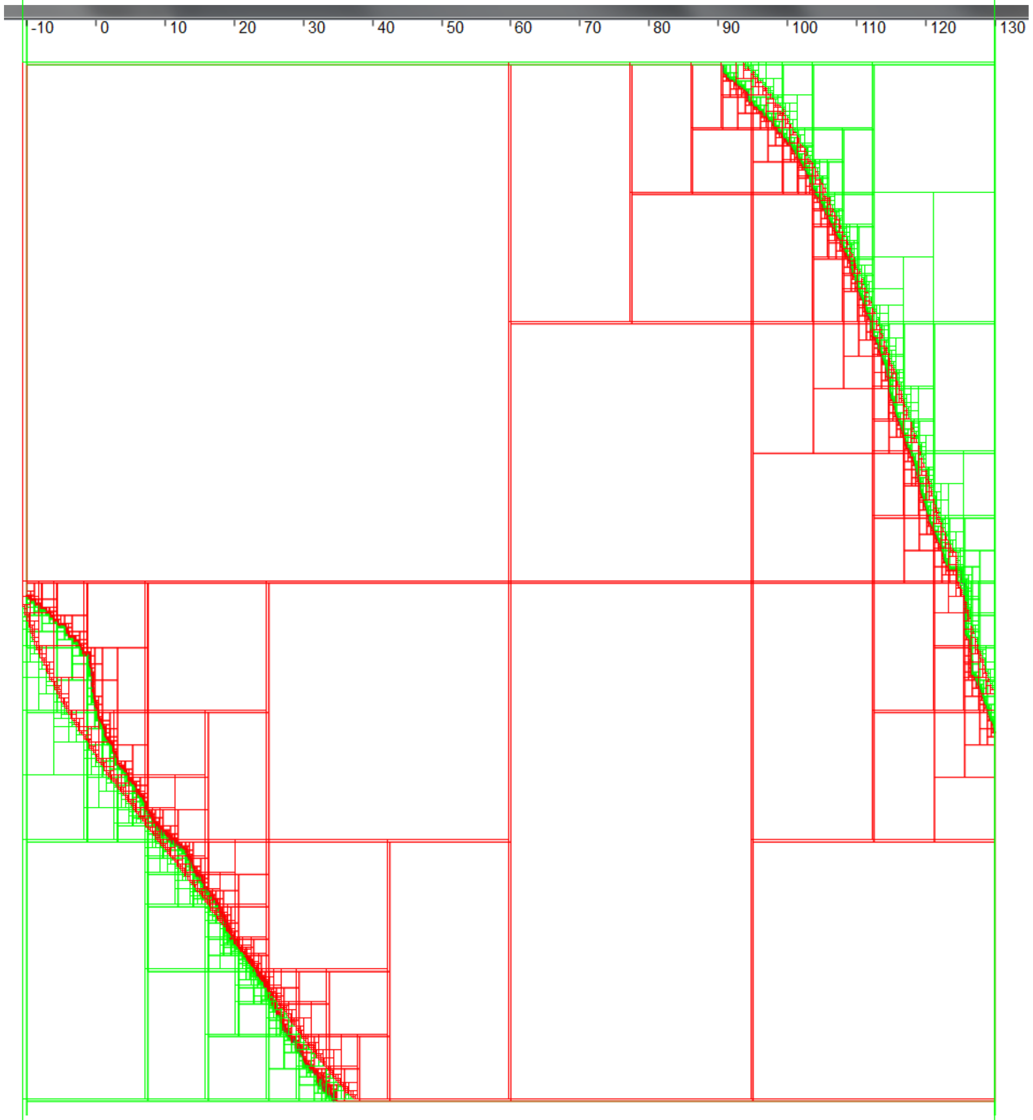


Viability Kernel – Contractor Based Algorithm
Mono Processor

$\varepsilon = 0.01 \ (m, box\ size)$
$dt = 0.01$ (sec, time step)
$nb\ iterations = 100$
$Euler$



Viability Kernel – Contractor Based Algorithm Mono Processor versus Bisection Based Algorithm (K1)



Synthesis, Conclusion and Way Forward

Synthesis

Real time viability kernel algorithm (K1, bisection)

Benchmark example : COTH

First performance features have been presented

Comparisons respect to contractor based approaches (mono processor version)

Conclusion

A lot of autonomous problems can be turned into viability questions

Way Forward

Capture basin algorithm (using bisections and contractors)

Differential games (problems involving two or more players / controls)

For more details about differential games, please refer to Professor Pierre Cardaliaguet, Eitan Altman …

Thank you !

Do you have questions ?

Spare Slides