The crazy fly

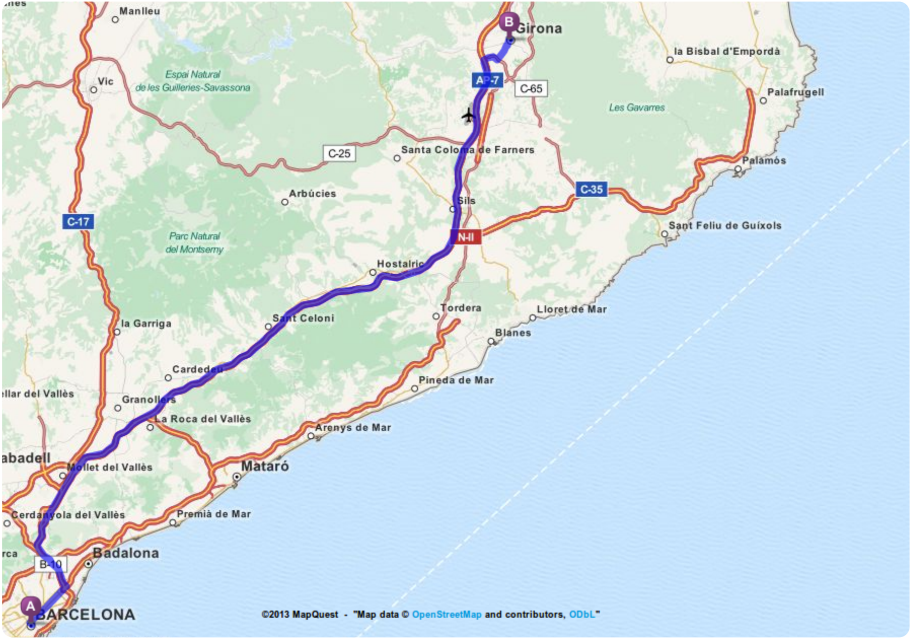
Timothy Bourke[1,2] Marc Pouzet[3,2,1]

1. INRIA Paris-Rocquencourt
2. École normale supérieure (DI)
3. Université Pierre et Marie Curie

`http://www.di.ens.fr/ParkasTeam.html`

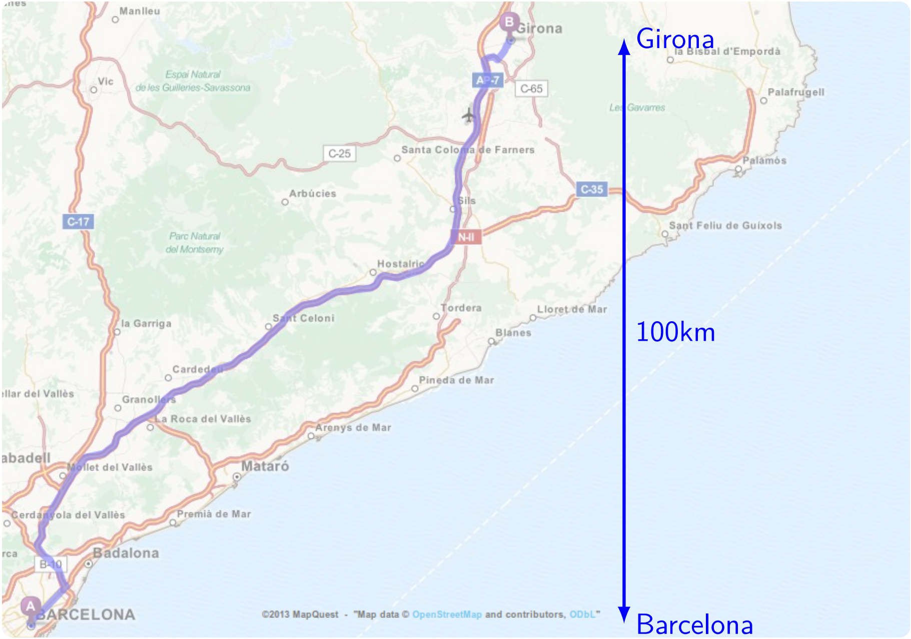
Synchron 2013, November 19, Daghstuhl, Germany

A very fast fly



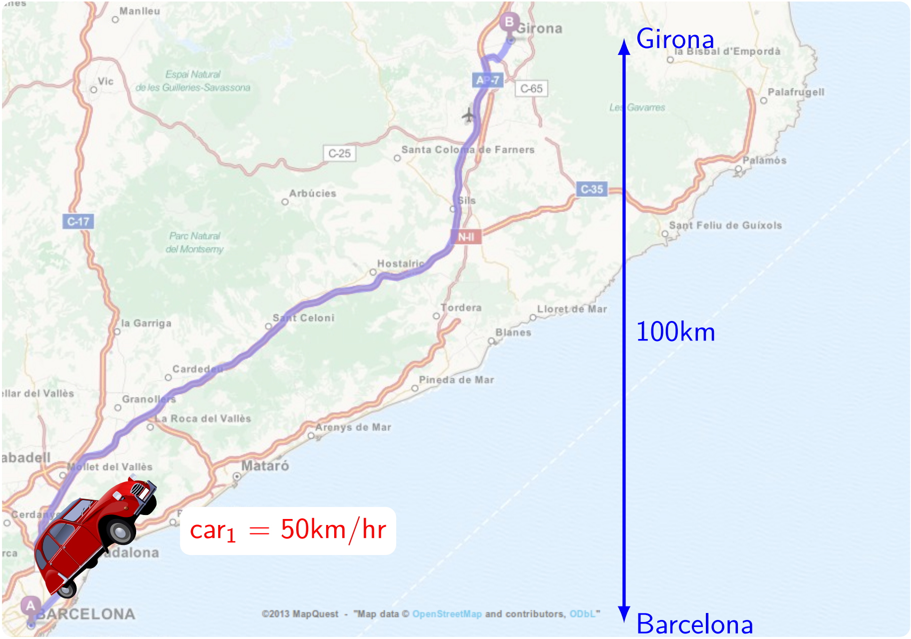
©2013 MapQuest - "Map data © OpenStreetMap and contributors, ODbL"

A very fast fly



©2013 MapQuest - "Map data © OpenStreetMap and contributors, ODbL"

A very fast fly



Girona

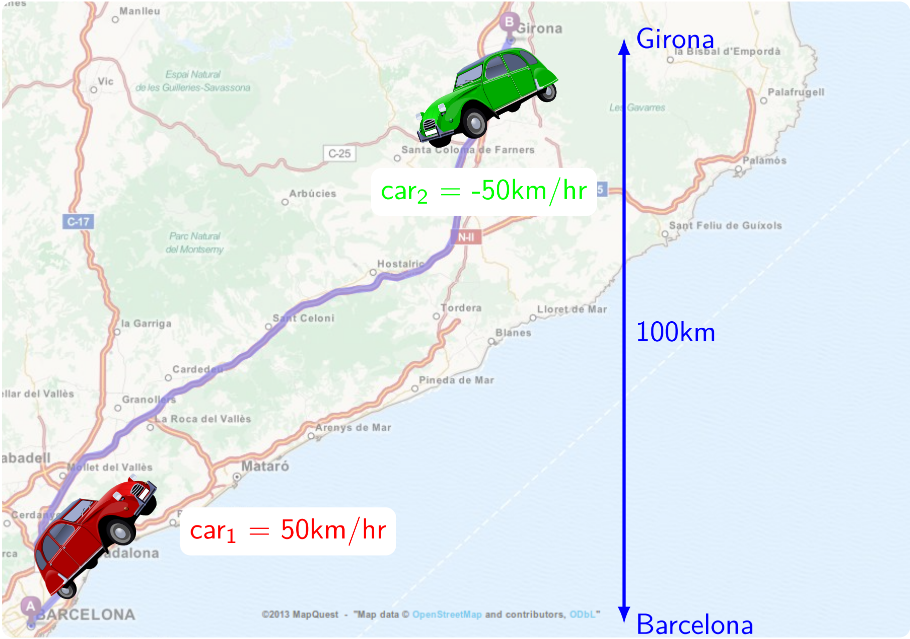
100km

Barcelona

$\text{car}_1 = 50\text{km/hr}$

©2013 MapQuest - "Map data © OpenStreetMap and contributors, ODbL"

A very fast fly



Girona

$\text{car}_2 = \text{-}50\text{km/hr}$

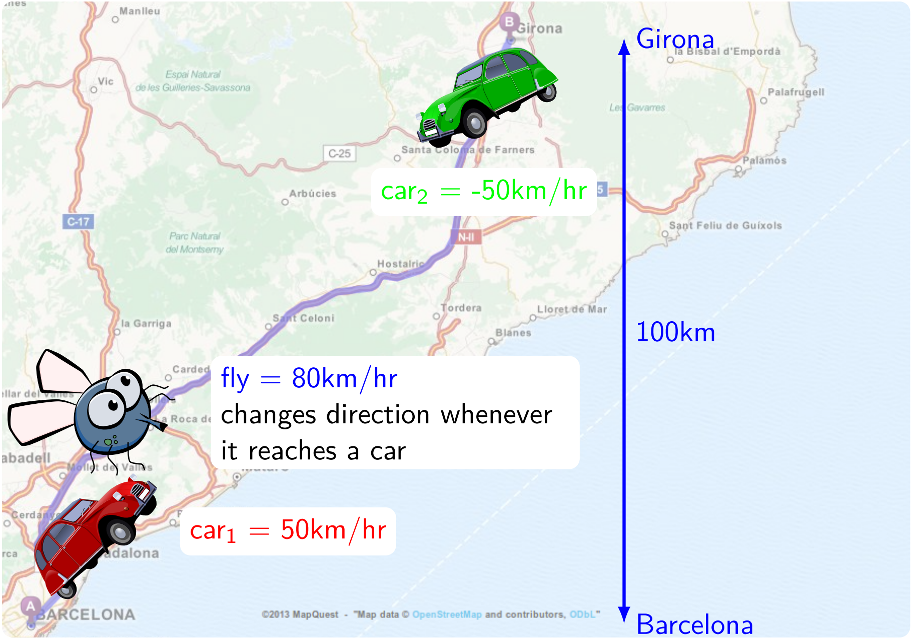
100km

$\text{car}_1 = 50\text{km/hr}$

Barcelona

©2013 MapQuest - "Map data © OpenStreetMap and contributors, ODbL"

A very fast fly



Girona

car_2 = -50km/hr

100km

fly = 80km/hr
changes direction whenever
it reaches a car

car_1 = 50km/hr

Barcelona

©2013 MapQuest - "Map data © OpenStreetMap and contributors, ODbL"

A very fast fly

The usual questions

1. How far has the fly traveled when the two cars meet?
2. How many zig-zags does the fly do during this period?

A very fast fly

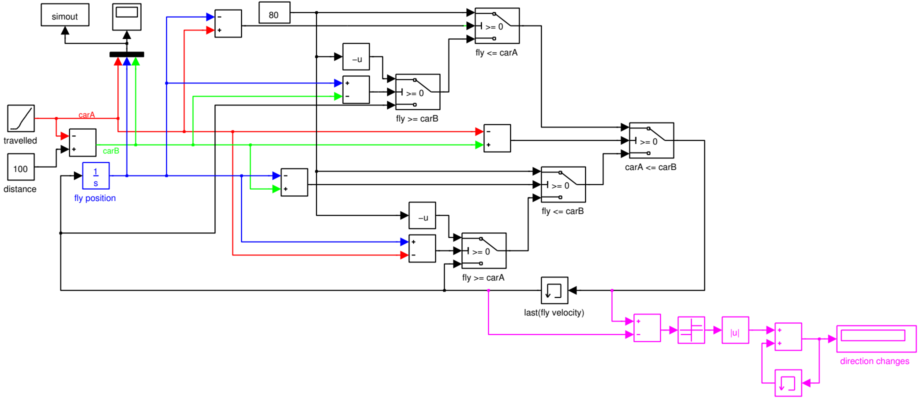
The usual questions

1. How far has the fly traveled when the two cars meet?
2. How many zig-zags does the fly do during this period?

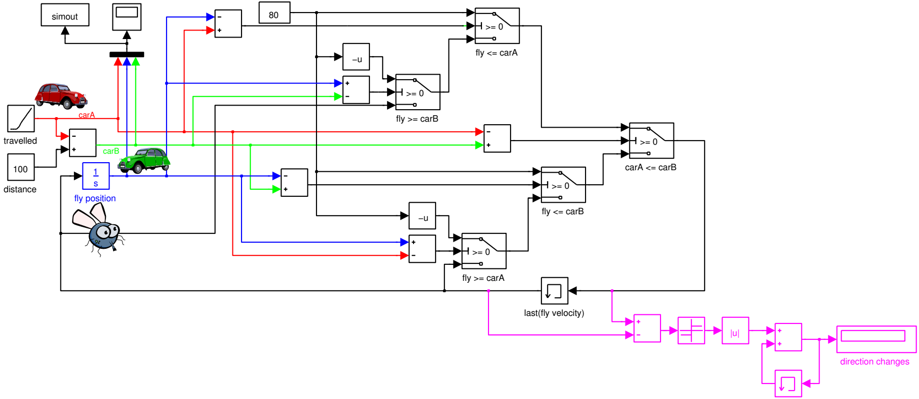
Extra credit (Thanks to Rafel Cases and Jordi Cortadella)

1. Where will the fly be when the two cars reach their destinations?

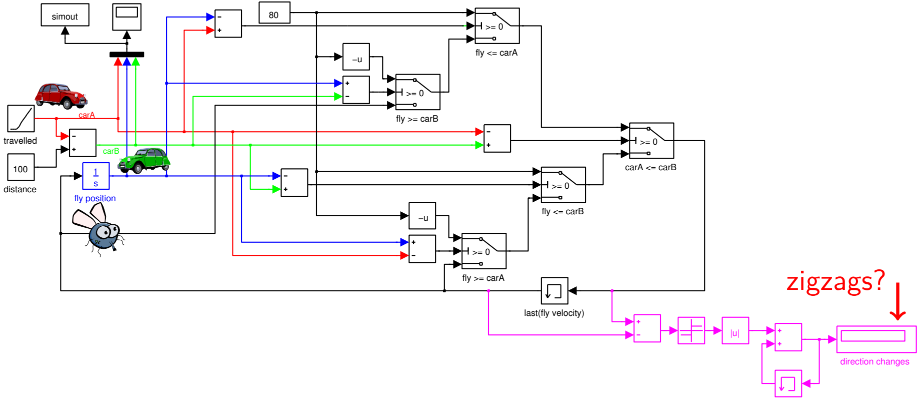
Simulink model



Simulink model

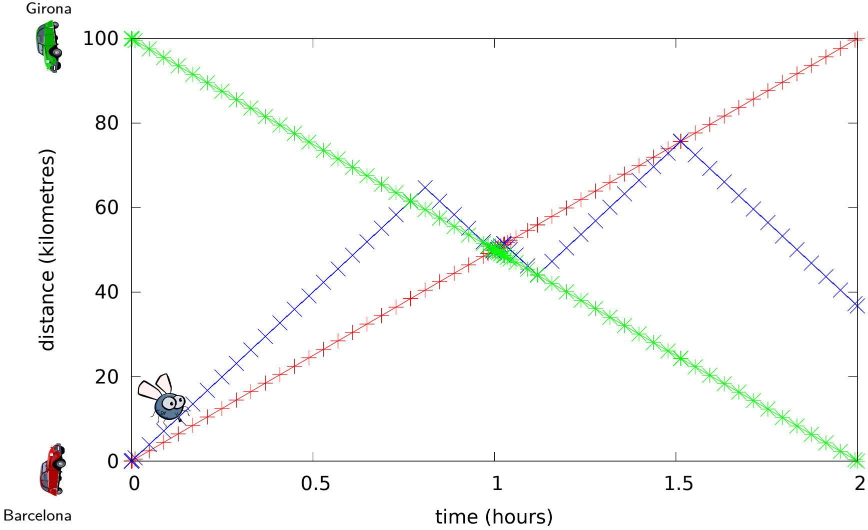


Simulink model



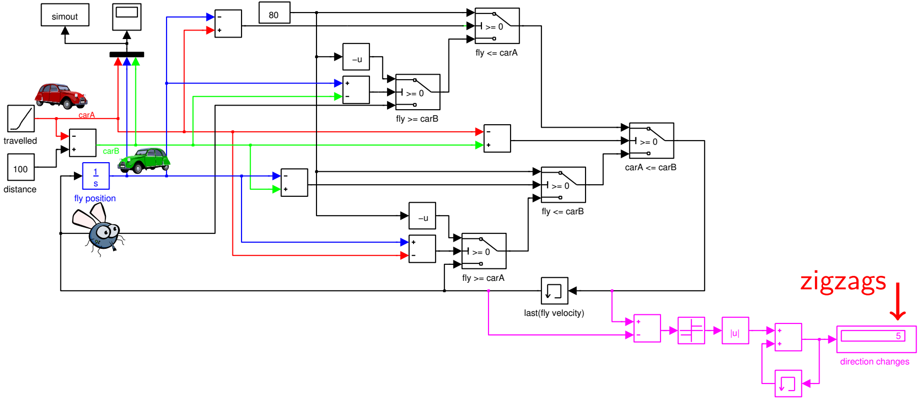
zigzags?

Simulink Results

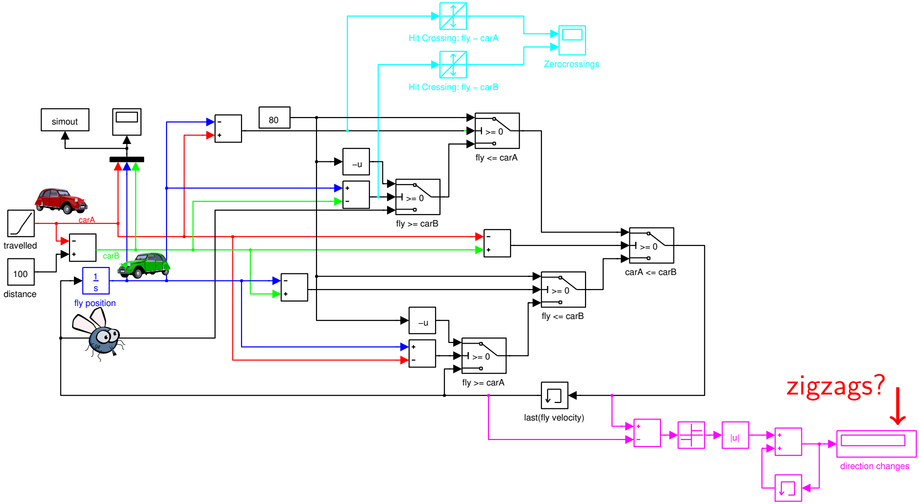


(Simulink R2012a: ode45, relative tolerance = 1e-3)

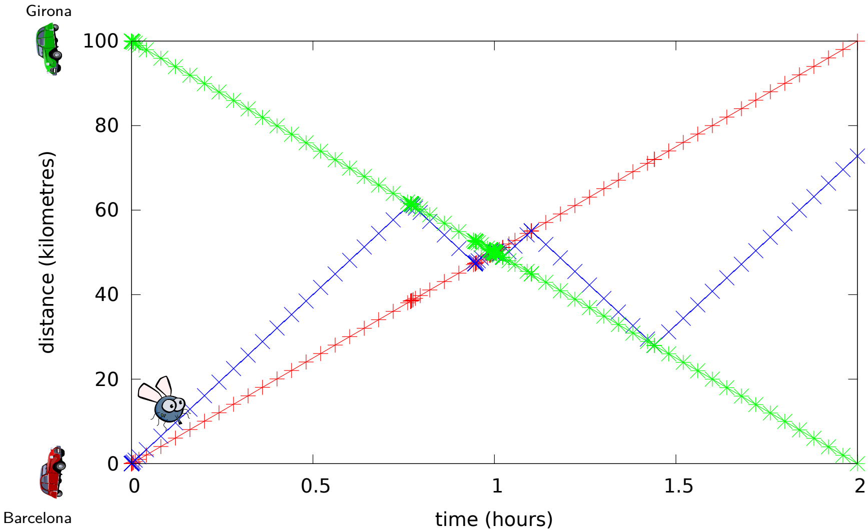
Simulink model



Simulink model (with more zero-crossings)

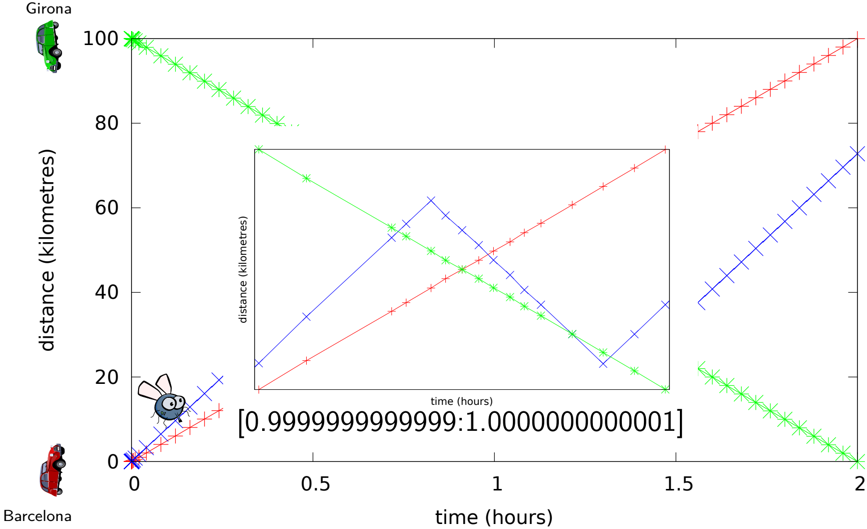


Simulink Results (with more zero-crossings)



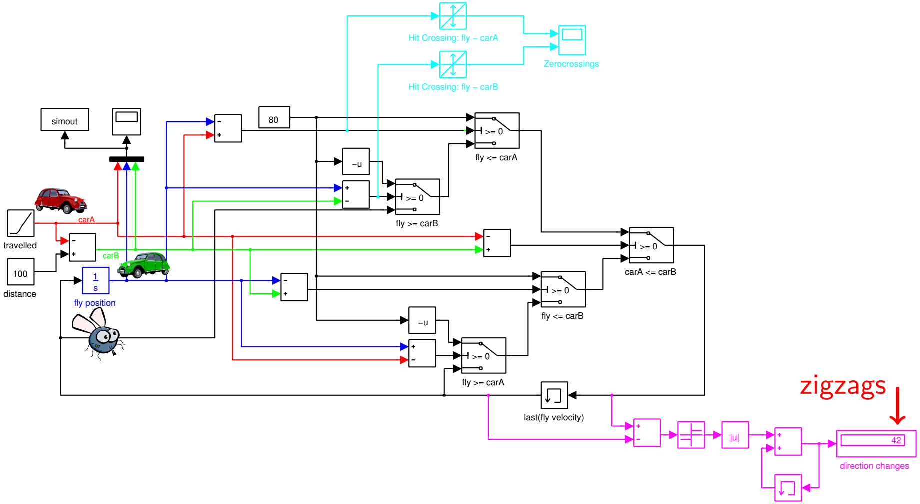
(Simulink R2012a: ode45, relative tolerance = 1e-3)

Simulink Results (with more zero-crossings)



(Simulink R2012a: ode45, relative tolerance = 1e-3)

Simulink model (with more zero-crossings)



42!

Let us try Zélus...

Zélus model [1]

```
let barcelona = 0.0
let girona = 100.0

let fly_velocity = 80.0
let car_velocity = 50.0

let hybrid model () =  (car1, car2, fly, zigzag, zeros) where
  rec der car1 = car_velocity      init barcelona
  and der car2 = −. car_velocity init girona
  and der fly = dir ∗. fly_velocity init barcelona
  and automaton
      | Above → (∗ the line above the fly ∗)
          do car_above = car2
          and car_below = car1
          until up(car1 −. car2) then Below
      | Below → (∗ the line below ∗)
          do car_above = car1
          and car_below = car2
          done
      end
  and present
       up (car_below −. fly) | up(fly −. car_above) →
        (∗ the fly changes her direction ∗)
        (∗ when she crosses the line below or the line above ∗)
        do dir = −. (last dir)
        and zeros = last zeros + 1
        and emit zigzag = ()
        done
  and init dir = 1.0
  and init zeros = 0
```

Zélus model [1]

```
let barcelona = 0.0
let girona = 100.0

let fly_velocity = 80.0
let car_velocity = 50.0

let hybrid model () =  (car1, car2, fly, zigzag, zeros) where
  rec der car1 = car_velocity      init barcelona
  and der car2 = −. car_velocity init girona
  and der fly = dir ∗. fly_velocity init barcelona
  and automaton
        | Above → (∗ the line above the fly ∗)
            do car_above = car2
            and car_below = car1
            until up(car1 −. car2) then Below
        | Below → (∗ the line below ∗)
            do car_above = car1
            and car_below = car2
            done
        end
  and present
         up (car_below −. fly) | up(fly −. car_above) →
           (∗ the fly changes her direction ∗)
           (∗ when she crosses the line below or the line above ∗)
           do dir = −. (last dir)
           and zeros = last zeros + 1
           and emit zigzag = ()
           done
  and init dir = 1.0
  and init zeros = 0
```

Zélus model [1]

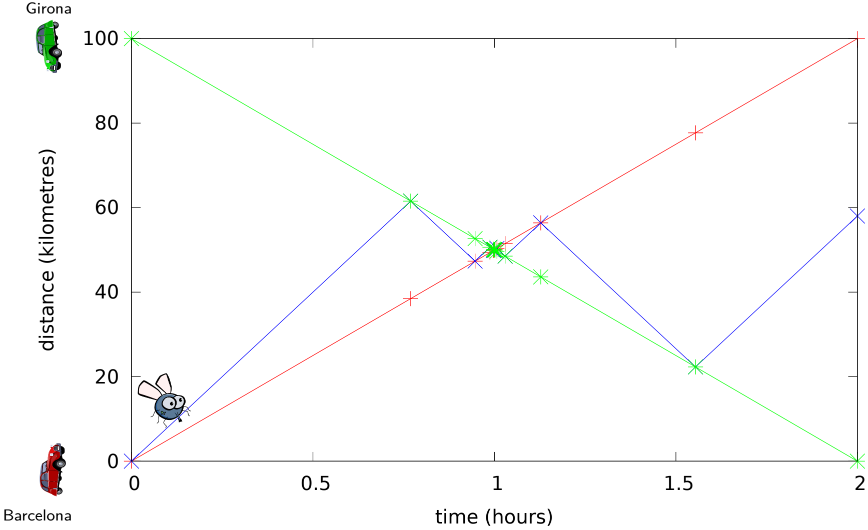
```
let barcelona = 0.0
let girona = 100.0

let fly_velocity = 80.0
let car_velocity = 50.0
```

zigzags=48

```
let hybrid model () =  (car1, car2, fly, zigzag, zeros) where
  rec der car1 = car_velocity      init barcelona
  and der car2 = -. car_velocity init girona
  and der fly = dir *. fly_velocity init barcelona
  and automaton
      | Above -> (* the line above the fly *)
          do car_above = car2
          and car_below = car1
          until up(car1 -. car2) then Below
      | Below -> (* the line below *)
          do car_above = car1
          and car_below = car2
          done
      end
  and present
       up (car_below -. fly) | up(fly -. car_above) ->
         (* the fly changes her direction *)
         (* when she crosses the line below or the line above *)
         do dir = -. (last dir)
         and zeros = last zeros + 1
         and emit zigzag = ()
         done
  and init dir = 1.0
  and init zeros = 0
```

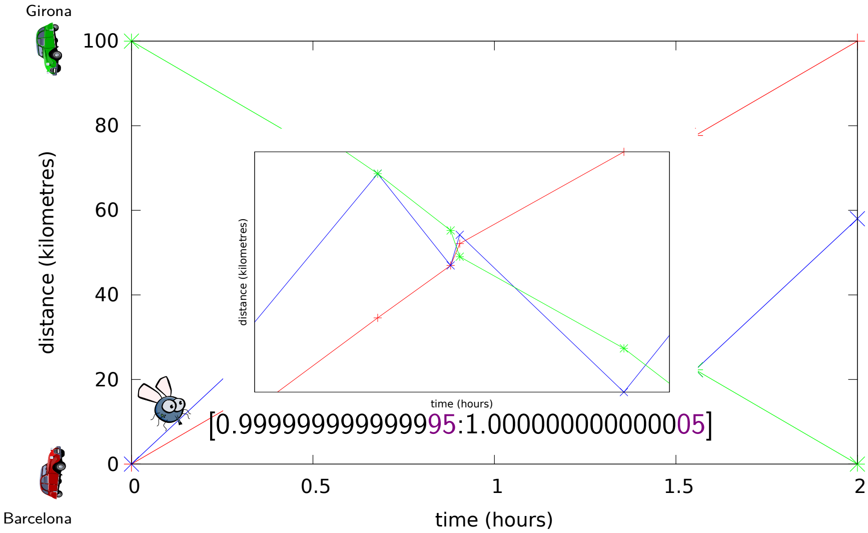
[1] zelus.di.ens.fr

Zélus Results



(Sundials CVODE with our custom Illinois implementation)

Zélus Results



(Sundials CVODE with our custom Illinois implementation)

Concluding remarks

Simulink

42 is the answer to "The Ultimate Question of Life, the Universe and Everything" [2]. Trop fort!

Well...

- All very well, but the problem is mathematically not well posed.
- The system is not well defined at the instant the cars pass each other.

Question: should a hybrid modeler

- statically detect and reject such programs?
- stop with an error at runtime?[3]

(Thanks to Rafel Cases, Jordi Cortadella, and Gérard Berry.)

[2] cf Douglas Adams, The Hitchhiker's Guide to the Galaxy.

[3] In the same way variable-step integration fails when reaching a minimal horizon.