A slow afternoon chez PARKAS and a very fast fly (our grand challenge)

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

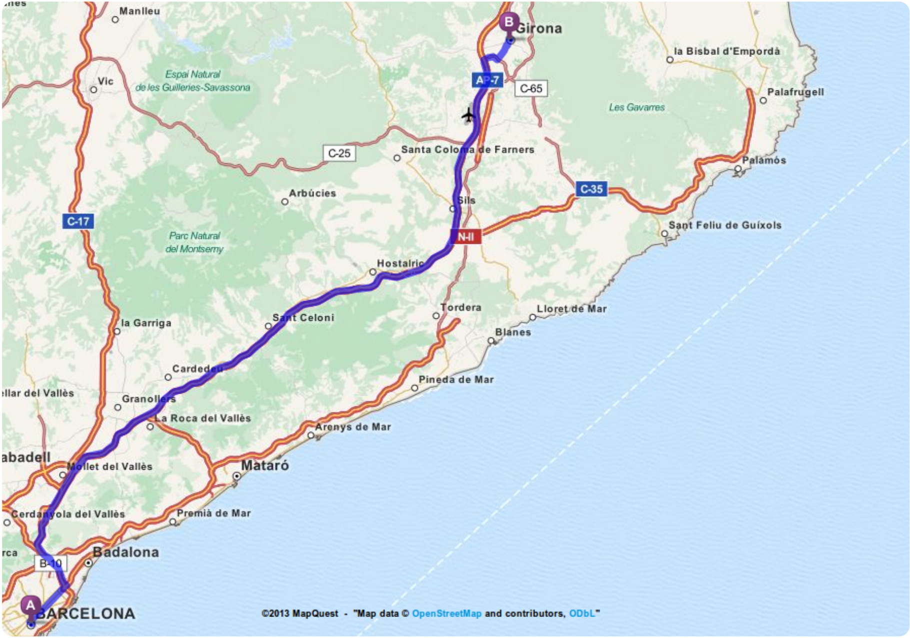
1. INRIA Paris-Rocquencourt

2. École normale supérieure (DI)

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

Synchron 2013, November 19, Daghstuhl, Germany

A very fast fly

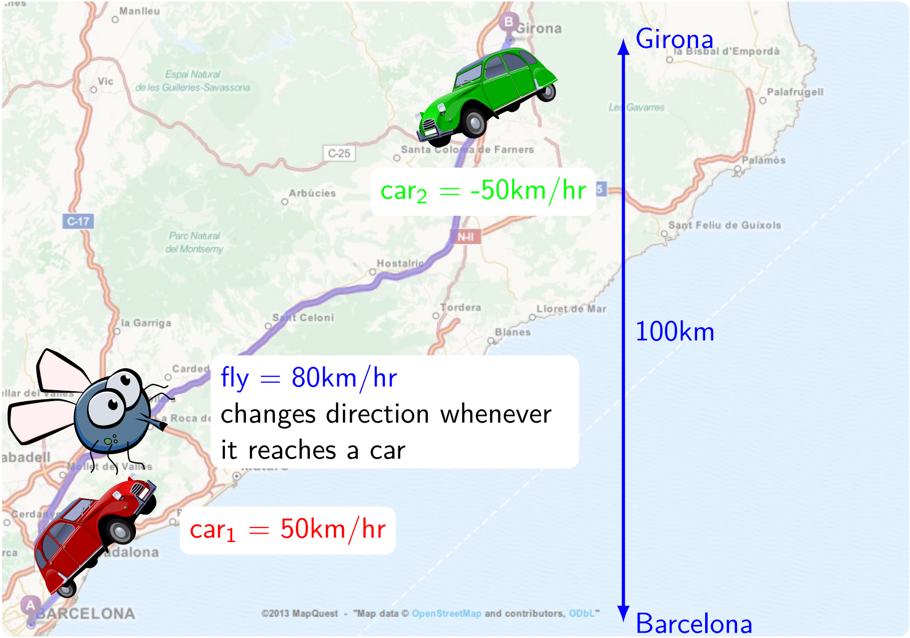


A very fast fly

A very fast fly

A very fast fly

A very fast fly



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

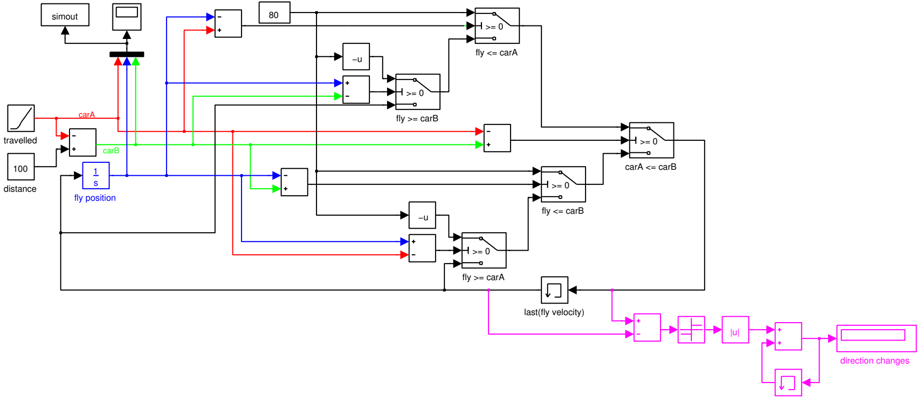
The usual questions

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

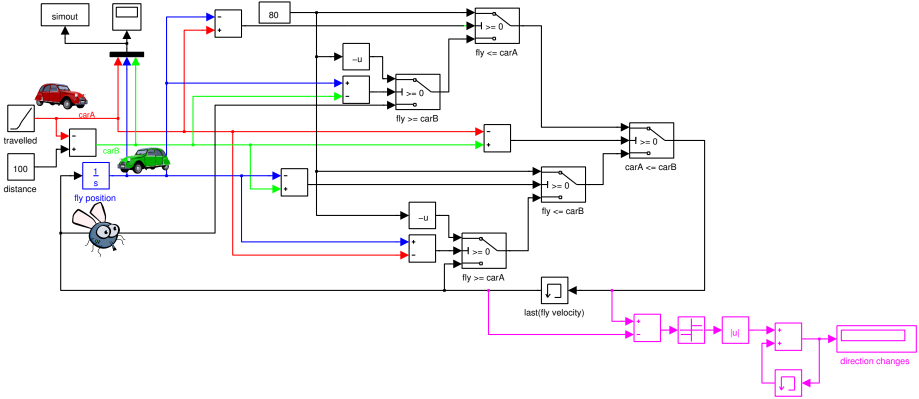
Extra credit (Thanks to Rafel Cases and Jordi Cortadella)

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

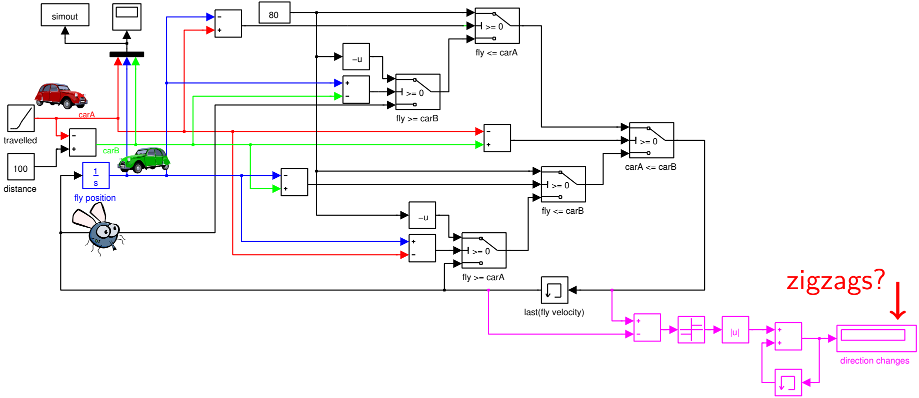
Simulink model



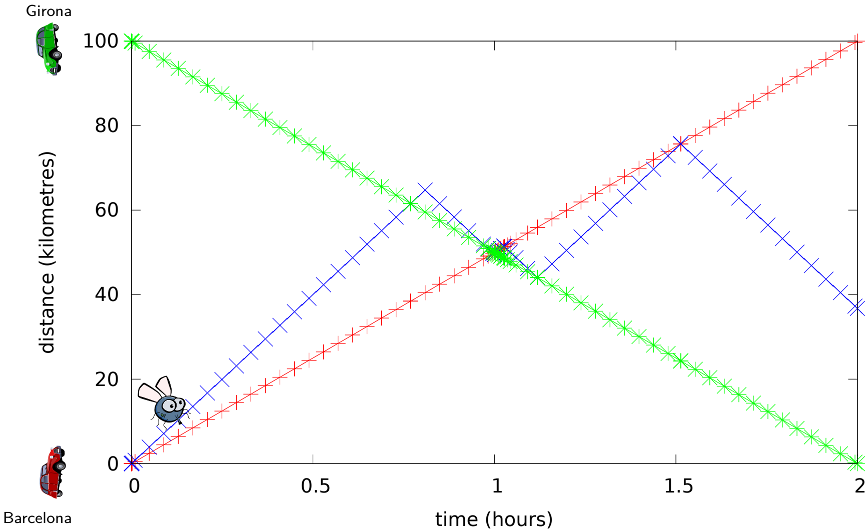
Simulink model



Simulink model

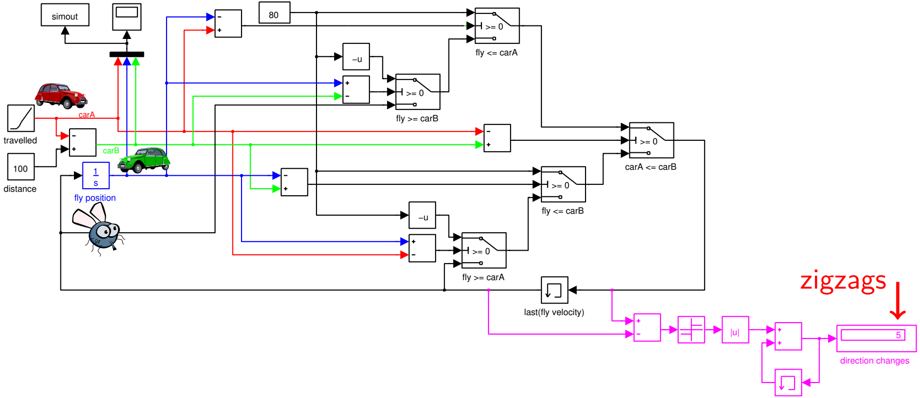


Simulink Results

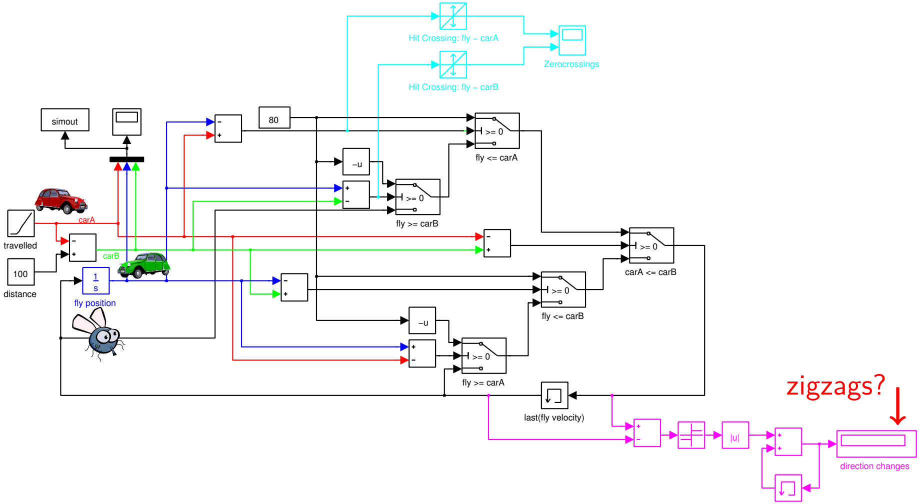


(Simulink R2012a: ode45, relative tolerance = 1e-3)

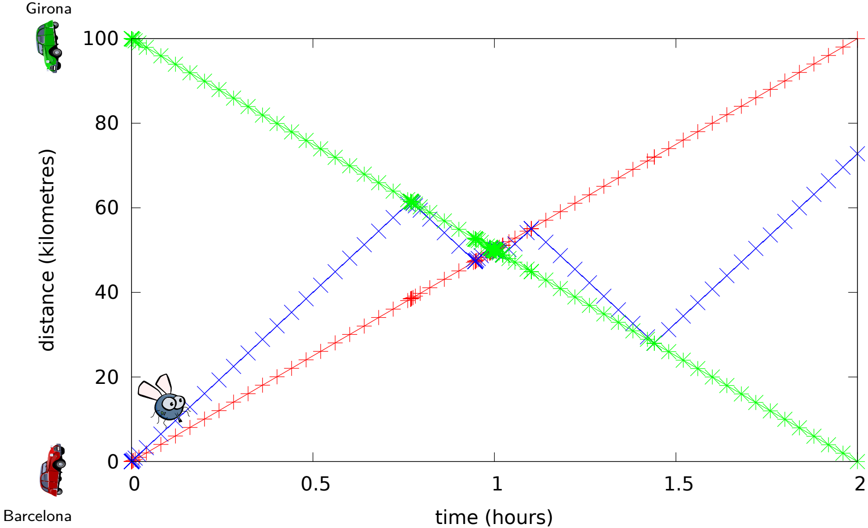
Simulink model



Simulink model (with more zero-crossings)

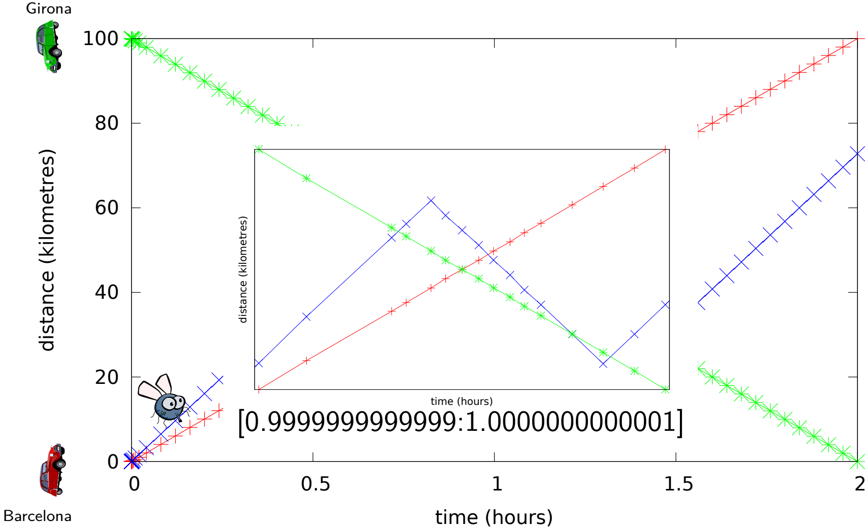


Simulink Results (with more zero-crossings)



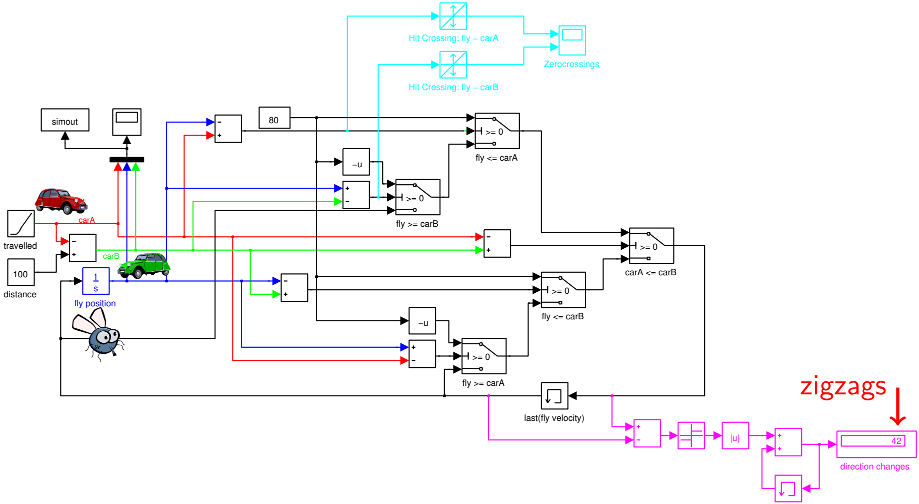
(Simulink R2012a: ode45, relative tolerance = 1e-3)

Simulink Results (with more zero-crossings)



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

Simulink model (with more zero-crossings)



42!

Zélus model

```
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 →
          do car_above = car2
          and car_below = car1
          until up(car1 −. car2) then Below
      | Below →
          do car_above = car1
          and car_below = car2
          done
      end
  and present
       up (car_below −. fly) | up(fly −. car_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



```
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 ->
          do car_above = car2
          and car_below = car1
          until up(car1 -. car2) then Below
      | Below ->
          do car_above = car1
          and car_below = car2
          done
      end
  and present
       up (car_below -. fly) | up(fly -. car_above) ->
        do
           dir = -. (last dir)
           and zeros = last zeros + 1
           and emit zigzag = ()
        done
  and init dir = 1.0
  and init zeros = 0
```

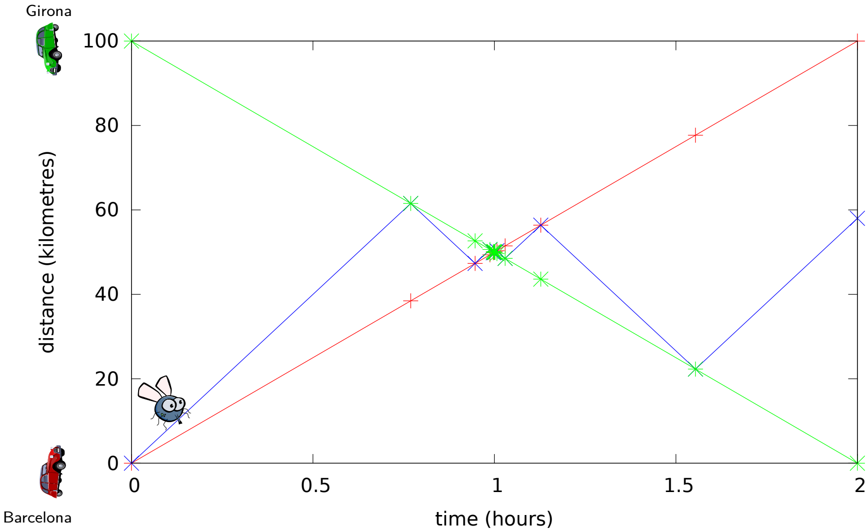
Zélus model

zigzags=48



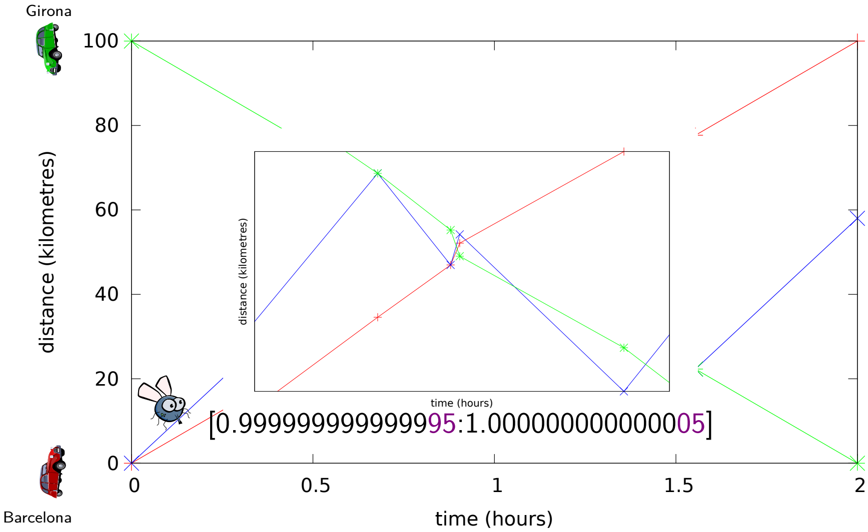
```
let barcelona = 0.0
let girona = 100.0

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 →
            do car_above = car2
            and car_below = car1
            until up(car1 -. car2) then Below
        | Below →
            do car_above = car1
            and car_below = car2
            done
        end
  and present
         up (car_below -. fly) | up(fly -. car_above) →
          do
             dir = -. (last dir)
             and zeros = last zeros + 1
             and emit zigzag = ()
          done
  and init dir = 1.0
  and init zeros = 0
```

Zélus Results



(Sundials CVODE with our custom Illinois implementation)

Zélus Results



(Sundials CVODE with our custom Illinois implementation)

Concluding remarks

- 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 we / can we:
  - statically detect and reject such cases?
  - stop with an error at runtime?
- (Thanks to Rafel Cases, Jordi Cortadella, and Gérard Berry.)