« Automatic program verification by Lagrangian relaxation and semidefinite programming »

Patrick Cousot
École normale supérieure
45 rue d'Ulm
75230 Paris cedex 05, France

Patrick.Cousot@ens.fr
www.di.ens.fr/~cousot

Semantics lunch — Cambridge, UK — Oct. 18th, 2004

Aperitif:
Relational semantics of loops

Relational semantics of loops

`while B do C od`

- $x \in \mathbb{R}/\mathbb{Q}/\mathbb{Z}$: values of the loop variables *before* a loop iteration
- $x' \in \mathbb{R}/\mathbb{Q}/\mathbb{Z}$: values of the loop variables *after* a loop iteration
- $[\![\mathtt{B};\mathtt{C}]\!](x, x')$: relational semantics of *one loop iteration*
- $[\![\mathtt{B};\mathtt{C}]\!](x, x') = \bigwedge_{i=1}^{N} \sigma_i(x, x') \geqslant 0$ (where $\geqslant$ is $>$, $\geq$ or $=$)
- not a restriction for numerical programs

Example of quadratic form program (factorial)

$$[x\,x']A[x\,x']^\top + 2[x\,x']\,q + r \geqslant 0$$

```
n := 0;                          -1.f +1.N >= 0
f := 1;                          +1.n >= 0
while (f <= N) do                +1.f -1 >= 0
    n := n + 1;                  -1.n +1.n' -1 = 0
    f := n * f                   +1.N -1.N' = 0
od                               -1.f.n' +1.f' = 0
```

$$[n\,f\,N\,n'\,f'\,N']\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} n \\ f \\ N \\ n' \\ f' \\ N' \end{bmatrix} + 2[n\,f\,N\,n'\,f'\,N']\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{2} \\ 0 \end{bmatrix} + 0 = 0$$

Appetiser:
Floyd/Hoare/Naur correctness proof method

Invariance proof

Given a loop precondition P, find an unkown loop invariant I such that:

– The invariant is *initial*:

$$\forall\, x : P(x) \Rightarrow I(x)$$

– The invariant is *inductive*:

$$\forall\, x, x' : I(x) \wedge [\![\mathtt{B};\mathtt{C}]\!](x, x') \Rightarrow I(x')$$

Invariance proof for numerical programs

Given a loop precondition $P(x) \geqslant 0$, find an unkown loop invariant $I(x) \geqslant 0$ such that:

– The invariant is *initial*:

$$\forall\ x : P(x) \geqslant 0 \Rightarrow I(x) \geqslant 0$$

– The invariant is *inductive*:

$$\forall\ x, x' : \left(I(x) \geqslant 0 \wedge \bigwedge_{i=1}^{N} \sigma_i(x, x') \geqslant 0 \right) \Rightarrow I(x') \geqslant 0$$

Termination proof

Given a loop invariant I, find an $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unkown rank function r such that:

– The rank is *nonnegative*:

$$\forall\ x : I(x) \Rightarrow r(x) \geq 0$$

– The rank is *strictly decreasing*:

$$\forall\ x, x' : I(x) \wedge [\![\mathtt{B};\mathtt{C}]\!](x, x') \Rightarrow r(x') \leq r(x) - \eta$$

$\eta = 1$ for $\mathbb{Z}$, $\eta > 0$ for $\mathbb{R}/\mathbb{Q}$ to avoid Zeno $\frac{1}{2}, \frac{1}{4}, \frac{1}{8} \ldots$

Wine service:

Iterated forward/backward static analysis for conditional termination

Conditional termination

– In general a loop does not terminate for all initial values of the variables

– In that case we can find no rank function!

– We must automatically determine a necessary loop precondition

– We use a iterated forward/backward static analysis ... with an auxiliary counter counting the number of remaining iterations down to zero

Arithmetic mean example, polyhedral abstraction without auxiliary counter)

```
{x>=y}
  while (x <> y) do
    {x>=y+2}
      x := x - 1;
    {x>=y+1}
      y := y + 1
    {x>=y}
  od
{x=y}
```

Arithmetic mean example, polyhedral abstraction with auxiliary counter

```
{x=y+2k,x>=y}
  while (x <> y) do
    {x=y+2k,x>=y+2}
      k := k - 1;
    {x=y+2k+2,x>=y+2}
      x := x - 1;
    {x=y+2k+1,x>=y+1}
      y := y + 1
    {x=y+2k,x>=y}
  od
{x=y,k=0}
  assume (k = 0)
{x=y,k=0}
```

Entrée:

Abstraction to parametric constraints

Parametric constraints

– Fix the form of the unkown ($I(x) \geqslant 0 / r(x) \geqslant 0$) using parameters a in the form $Q(a, x) \geqslant 0$

– This is an abstraction

– Examples:

- $r(x, y) = a.x + b.y + c$
- $I(x, x') = a.x^2 + b.x.x' + c.x'^2 + d.x + e.x' + f$

Solving the constraints

– The invariance [termination] problems have the form:

$$\exists\, a : \forall\, x, x' :$$
$$\left([Q(a, x) \geqslant 0 \wedge] \bigwedge_{k=1}^{n} C_k(x, x') \geqslant 0 \right)$$
$$\Rightarrow$$
$$Q'(a, x, x') \geqslant 0$$

– Find an algorithm to effectively compute a!

Problems

In order to compute a:

– How to handle $\bigwedge$?

– How to get rid of the implication $\Rightarrow$?

$\longrightarrow$ Lagrangian relaxation

– How to get rid of the universal quantification $\forall$?

– How to handle $\wedge$?

$\longrightarrow$ quantifier elimination (does not scale up)

$\longrightarrow$ mathematical programming

Algorithmically interesting cases

– linear inequalities

→ linear programming [1]

– linear matrix inequalities (LMI)/quadratic forms

– bilinear matrix inequalities (BMI)

→ semidefinite programming

– semialgebraic sets

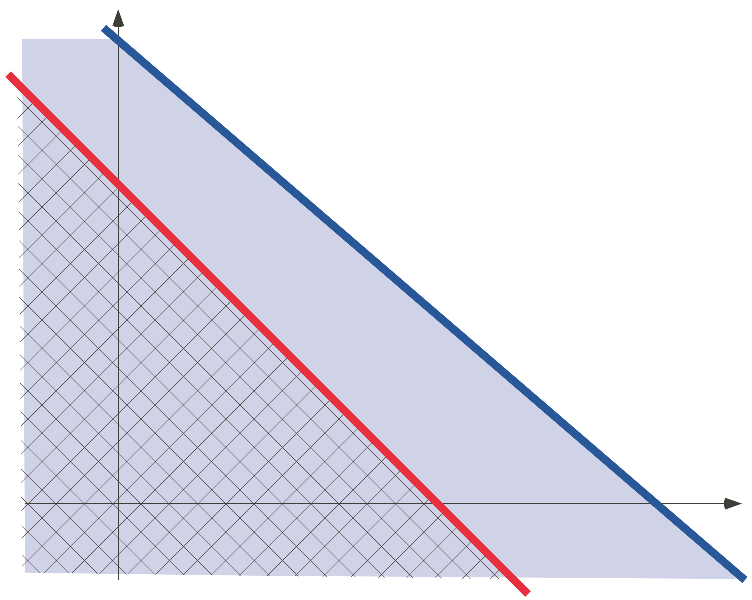
→ polynomial quantifier elimination, or

→ relaxation with semidefinite programming

[1] Already explored for invariants by Sankaranarayanan, Spima, Manna (CAV'03, SAS'04, heuristic solver) and for termination by Podelski & Rybalchenko (VMCAI'03, Lagrange coefficients eliminated by hand to reduce to linear programming so no disjunctions, no tests, etc).

First main course: Lagrangian relaxation for implication elimination

Example of linear Lagrangian relaxation



$$A \Rightarrow B \qquad\qquad \text{(assum ng } A \neq \emptyset)$$

$\Leftarrow$ (soundness)

$\Rightarrow$ (completeness)

border of A parallel to border of B

Lagrangian relaxation, formally

Let $\mathbb{V}$ be a finite dimensional linear vector space, $N > 0$ and $\forall k \in [1, N] : \sigma_k \in \mathbb{V} \mapsto \mathbb{R}$.

$$\forall x \in \mathbb{V} : \left(\bigwedge_{k=1}^{N} \sigma_k(x) \geq 0 \right) \Rightarrow (\sigma_0(x) \geq 0)$$

$\Leftarrow$ soundness (Lagrange)

$\Rightarrow$ completeness (*lossless*)

$\not\Rightarrow$ ncompleteness (*lossy*)

$$\exists \lambda \in [1, N] \mapsto \mathbb{R}_* : \forall x \in \mathbb{V} : \sigma_0(x) - \sum_{k=1}^{N} \lambda_k \sigma_k(x) \geq 0$$

relaxation = approximation, λ_i = Lagrange coefficients

Lagrangian relaxation, completeness cases

- Linear case
  (affine Farkas' lemma)
- Linear case with at most 2 quadratic constraints
  (Yakubovich's S-procedure)

Lagrangian relaxation of the constraints

$$\exists\ a : \forall\ x, x' : [Q(a, x) \geqslant 0 \wedge] \bigwedge_{k=1}^{n} C_k(x, x') \geqslant 0$$

$$\Rightarrow Q'(a, x, x') \geqslant 0$$

$\Leftarrow$ (is *relaxed* into)

$$\exists\ a : [\exists \lambda \geqslant 0] : \exists \lambda_k \geqslant 0 : \forall\ x, x' :$$

$$Q'(a, x, x')[-\lambda.Q(a, x)] - \sum_{k=1}^{n} \lambda_k.C_k(x, x') \geqslant 0$$

$\uparrow$ linear in a $\qquad\qquad$ $\uparrow$ linear in the λ_k

$\uparrow$ bilinear in a & λ

Second main course: Mathematical programming for quantifier elimination

Mathematical programming

$$\exists x \in \mathbb{R}^n: \qquad \bigwedge_{i=1}^{N} g_i(x) \geqslant 0$$

$$[\text{Minimizing} \quad f(x)]$$

feasibility problem : find a solution to the constraints

optimization problem : find a solution, minimizing $f(x)$

Semidefinite programming

$$\exists x \in \mathbb{R}^n: \qquad M(x) \succcurlyeq 0$$

$$[\text{Minimizing} \quad cx]$$

Where the linear matrix inequality is

$$M(x) = M_0 + \sum_{k=1}^{n} x_k M_k$$

with symetric matrices ($M_k = {M_k}^\top$ and the positive semidefiniteness is

$$M(x) \succcurlyeq 0 = \forall X \in \mathbb{R}^N : X^\top M(x) X \geq 0$$

Semidefinite programming, once again

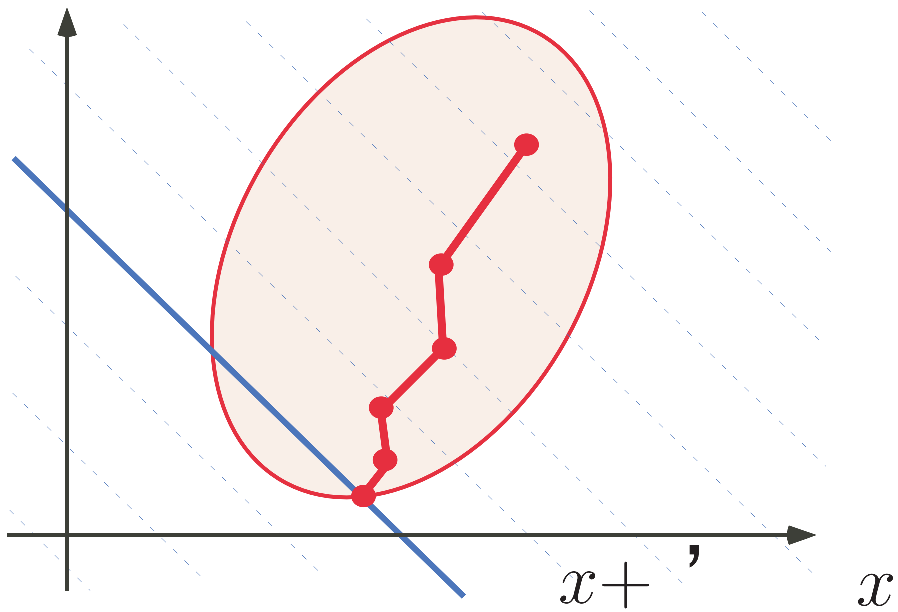
Feasibility is:

$$\exists x \in \mathbb{R}^n : \forall X \in \mathbb{R}^N : X^\top \left(M_0 + \sum_{k=1}^{n} x_k M_k \right) X \geq 0$$

of the form of the (linear) formulæ we are interested in for programs with linear matricial semantics.

Interior point method for semidefinite programming

– Nesterov & Nemirovskii 1988, polynomial in worst case and good in practice (thousands of variables)



– Various path strategies e.g. "stay in the middle"

Semidefinite programming solvers

Numerous solvers available under MATHLAB®, a.o.:

– `lmilab`: P. Gahinet, A. Nemirovskii, A.J. Laub, M. Chilali

– `Sdplr`: S. Burer, R. Monteiro, C. Choi

– `Sdpt3`: R. Tütüncü, K. Toh, M. Todd

– `SeDuMi`: J. Sturm

– `bnb`: J. Löfberg (integer semidefinite programming)

Common interfaces to these solvers, a.o.:

– `Yalmip`: J. Löfberg

Sometime need some help (feasibility radius, shift,...)

Recent generalization to bilinear matrix inequalities

– penbmi: M. Kočvara, M. Stingl

Feasibility is:

$$\exists x \in \mathbb{R}^n : \forall X \in \mathbb{R}^N :$$
$$X^\top \left(M_0 + \sum_{j=1}^{n} x_j M_j + \sum_{k=1}^{n} \sum_{\ell=1}^{n} x_k x_\ell M'_{k\ell} \right) X \geq 0$$

of the form of the (bilinear) formulæ we are interested in!

Skipping the cheese . . .

Not enough time for . . .

– Disjunctions in the loop test?
– Conditionals in the loop body?
– Nested loops?
– Concurrency?
– Fair parallelism?
– Semi-algebraic/polynomial programs?
– Data structures?

Desert
Invariance and Termination Examples

Termination of a linear program

```
{y >= 1}
while (x >= 1) do
  x := x - y
od
```

⟵ termination precondition determined by iterated forward/backward polyhedral analysis

`lmilab`:

`r(x,y) = +2.178955e+12.x +1.453116e+12.y -1.451513e+12`

`lmilab` (with feasibility radius of `1.0e4`):

`r(x,y) = +4.074723e+03.x +2.786715e+03.y +1.549410e+03`

`sedumi`:

`r(x,y) = +2.271450e+03.x +1.810903e+03.y -3.623997e+03`

`bnb` (integer semidefinite programming)[2]: `r(x,y) = +2.x+2.y-3`

[2] still in infancy!

Termination of the arithmetic mean

```
{x=y+2k,x>=y}
while (x <> y) do
  k := k - 1;
  x := x - 1;
  y := y + 1
od
{assert (k = 0)}
```

$\longleftarrow$ termination precondition determined by iterated forward/backward polyhedral analysis

`lmilab`:

```
r(x,y,k) = +1.382113e+03.x -1.382113e+03.y +4.978695e+03.k
                                              +2.711732e+03
```

Termination of the Euclidean division

```
1:  {y>=1}
  q := 0;
2:  {q=0,y>=1}
  r := x;
3:  {x=r,q=0,y>=1}
  while (y <= r) do
    4:  {y<=r,q>=0}
      r := - y + r;
    5:  {r>=0,q>=0}
      q := q + 1
    6:  {r>=0,q>=1}
  od
7:  {q>=0,y>=r+1}
```

⟵ termination precondition determined by iterated forward/backward polyhedral analysis

bnb:

```
r(y,q,r) = -2.y +2.q +4.r
```

Floyd's proposal $r(x, y, q, r) = x - q$ is more intuitive but requires to discover the nonlinear loop invariant $x = r + qy$.

Termination of a quadratic program: factorial

```
{true}
n := 0;
f := 1;
while (f <= N) do
  n := n + 1;
  f := n * f
od
```

⟵ termination precondition determined by iterated forward/backward polyhedral analysis

`sedumi` (with feasibility radius of `1.0e+3`):

```
r(n,f,N) = -9.993462e-01.n +1.617225e-04.f +2.688476e+02.N
                                          +8.745232e+02
```

Loop body with tests

```
while (x < y) do
  if (i >= 0) then
      x := x+i+1
  else
      y := y+i
  fi
od
```

lmilab:

```
r(i,x,y) = -2.252791e-09.i -4.355697e+07.x +4.355697e+07.y
                                              +5.502903e+08
```

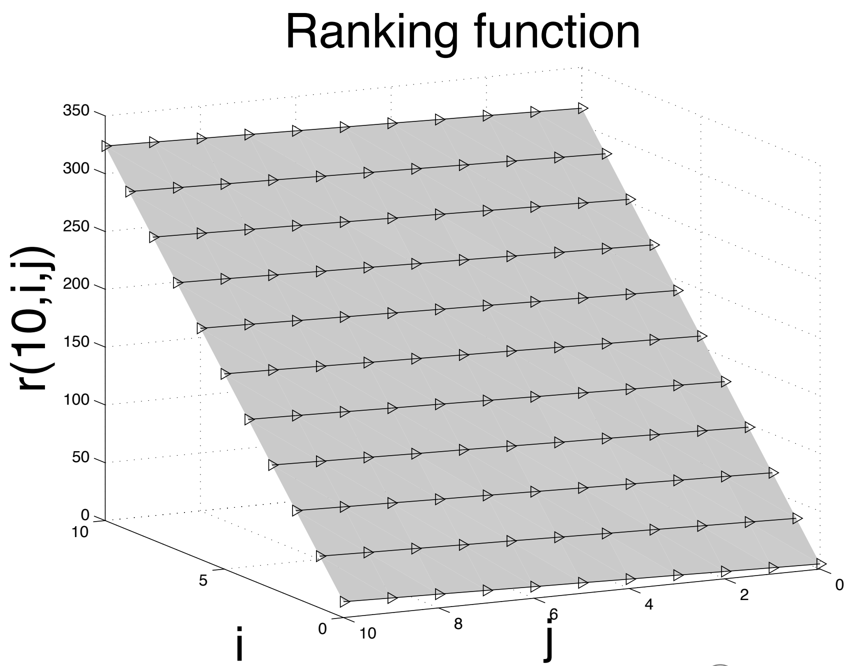
Quadratic termination of linear loop

```
{n>=0}
i := n; j := n;
while (i <> 0) do
  if (j > 0) then
    j := j - 1
  else
    j := n; i := i - 1
  fi
od
```

⟵ termination precondition determined by iterated forward/backward polyhedral analysis sdplr (with feasibility radius of 1.0e+3):

```
r(n,i,j) = +7.024176e-04.n^2 +4.394909e-05.n.i ...
           -2.809222e-03.n.j +1.533829e-02.n ...
           +1.569773e-03.i^2 +7.077127e-05.i.j   ...
           +3.093629e+01.i -7.021870e-04.j^2 ...
           +9.940151e-01.j +4.237694e+00
```

Successive values of $\texttt{r}(\texttt{n}, \texttt{i}, \texttt{j})$ for $\texttt{n} = 10$ on loop entry



Termination of a concurrent program

```
[|   1: while [x+2 < y] do
     2:      [x := x + 1]
        od
     3:
||
     1: while [x+2 < y] do
     2:      [y := y - 1]
        od
     3:
|]
```

interleaving $\longrightarrow$

```
while (x+2 < y) do
   if ?=0 then
     x := x + 1
   else if ?=0 then
     y := y - 1
   else
     x := x + 1;
     y := y - 1
   fi fi
od
```

penbmi: `r(x,y) = 2.537395e+00.x+-2.537395e+00.y+ -2.046610e-01`

Termination of a fair parallel program

```
[[ while [(x>0)|(y>0) do x := x - 1] od ||
   while [(x>0)|(y>0) do y := y - 1] od ]]
```

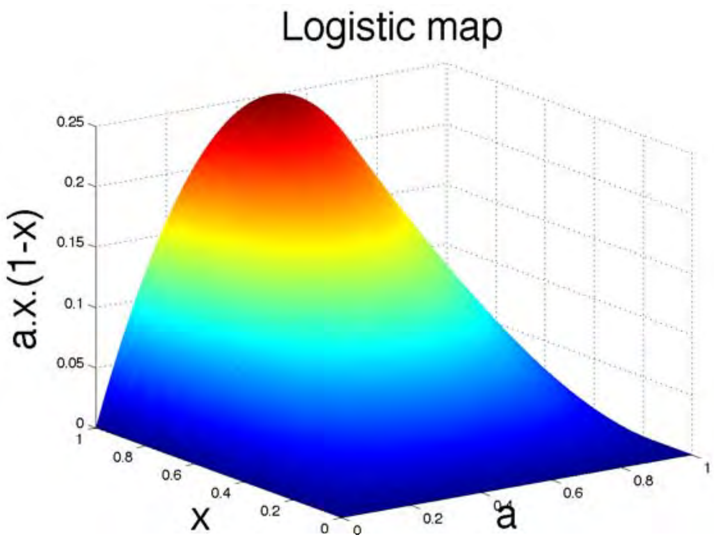
interleaving
+ scheduler
$\longrightarrow$

```
{m>=1}  ←  termination precondition determined by iterated
           forward/backward polyhedral analysis
t := ?;
assume (0 <= t & t <= 1);
s := ?;
assume ((1 <= s) & (s <= m));
while ((x > 0) | (y > 0)) do
  if (t = 1) then
     x := x - 1
  else
     y := y - 1
  fi;
  s := s - 1;
  if (s = 0) then
    if (t = 1) then
       t := 0
    else
       t := 1
    fi;
    s := ?;
    assume ((1 <= s) & (s <= m))
  else
    skip
  fi
od;;
```

penbmi: `r(x,y,m,s,t) = +1.000468e+00.x +1.000611e+00.y +2.855769e-02.m -3.929197e-07.s +6.588027e-06.t +9.998392e+03`

Semidefinite programming relaxation for polynomial programs

```
eps = 1.0e-9;
while (0 <= a) & (a <= 1 - eps)
          & (eps <= x) & (x <= 1) do
   x := a*x*(1-x)
od
```



Write the verification conditions in polynomial form, use SOStool to relax in semidefinite programming form.

SOStool+SeDuMi:

```
r(x) = 1.222356e-13.x + 1.406392e+00
```

When constraint resolution fails. . .

Infeasibility of the constraints does not mean "non termination" but simply failure:

– There can be a rank function of a different form (e.g. quadratic while looking for a linear one),

– The solver may have failed (e.g. add a shift).

Coffee: Conclusion

Numerical errors

– LMI solvers do numerical computations with rounding errors, shifts, etc
– rank function is subject to numerical errors
– the hard point is to discover a candidate for the rank function
– much less difficult, when it is known, to re-check for satisfaction (e.g. by static analysis)

Invariance for Euclidian division

```
assume (y > 0);
q := 0;
r := x;
while (y <= r) do
    r := - y + r;
    q := q + 1
od
```

yalmip bmi:

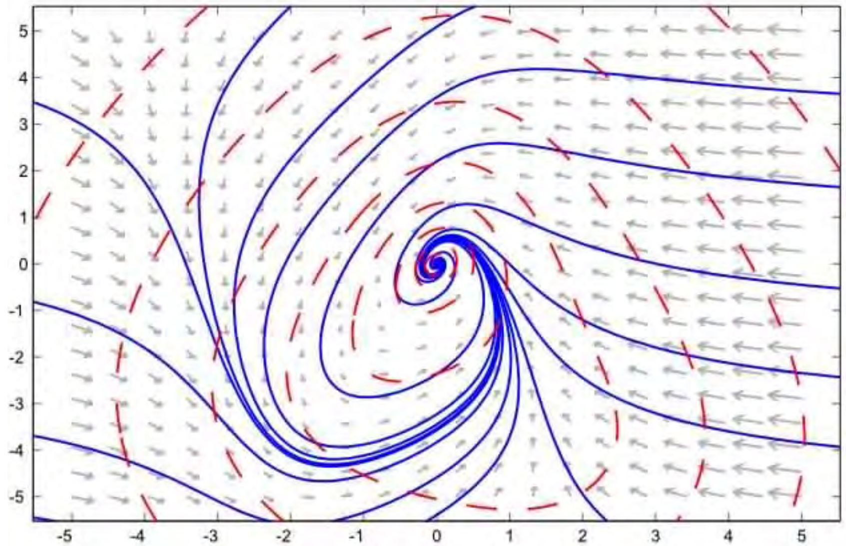
`1.337645e-04*x+2.484973e-04*q*y+1.588933e-03*r >= 0`

which is not false!

Digestif: Questions

Seminal work

– LMI case, Lyapunov 1890, *"an invariant set of a differential equation is stable in the sense that it attracts all solutions if one can find a function that is bounded from below and decreases along all solutions outside the invariant set"*.



THE END

I hope you had a good and *relaxed* semantics lunch