This Maple worksheet documents the calculations described in Section 4.2 for System 26 of the following paper.

Matthew England, Hassan Errami, Dima Grigoriev, Ovidiu Radulescu, Thomas Sturm, and Andreas Weber. Symbolic Versus Numerical Computation and Visualization of Parameter Regions for Multistationarity of Biological Networks. In Proceedings of CASC '17, Beijing, China, September 18 -22 2017, 15 pages. Springer, 2017.

Note - we are running in Maple 2016 with the latest version of the RegularChains linrary downloaded from www.regularchains.org

Date Written: 13th June 2017
Author: Matthew England (Coventry University)
Matthew.England@coventry.ac.uk

```
> restart:
> libname := "C:/Users/ab9797/Dropbox/Maple_CAD/FromRCorg/Latest",
  libname;
```

libname := "C:/Users/ab9797/Dropbox/Maple_CAD/FromRCorg/Latest",
"C:\Program Files\Maple 2016\lib"

```
> with(RegularChains):
> with(SemiAlgebraicSetTools);
```

[*BoxValues, Complement, CylindricalAlgebraicDecompose, Difference, DisplayParametricBox, DisplayQuantifierFreeFormula, EmptySemiAlgebraicSet, Intersection, IsContained, IsEmpty, IsParametricBox, LinearSolve, PartialCylindricalAlgebraicDecomposition, PositiveInequalities, Projection, QuantifierElimination, RealRootCounting, RealRootIsolate, RefineBox, RefineListBox, RemoveRedundantComponents, RepresentingBox, RepresentingChain, RepresentingQuantifierFreeFormula, RepresentingRootIndex, SignAtBox, VariableOrdering*]

Funnctions to iterate the grid sampling

```
> generate := proc(K17R, K18R, K19R, ansTab, TS, vars, update:=
  false)
  local check, r, K17vals, K18vals, K19vals, ans, i:

  for r in [K17R, K18R, K19R] do
    check := (type(r, list) and nops(r)=3 and r[2]>r[1]) or type
  (r, integer):
    if check<>true then
      error("First 3 arguments must each be an integer or a list
  of 3 integers (start, stop, jump)"):
    fi:
  od:

  if type(ansTab, table)<>true then
    error("4th argument must be a table"):
  fi:

  if type(TS, list)<>true then
    error("5th argument must be a list of polynomial
  constraints"):
  fi:

  if type(vars, list)<>true then
    error("6th argument must be a list of variables"):
  fi:

  if not( type(update,integer) or is(update=false) ) then
    error("Optional 7th argument must be an integer")
  fi:

  if type(K17R, integer) then
      K17vals:= [K17R]:
  else
      K17vals := []:
      for i from K17R[1] by K17R[3] to K17R[2] do
        K17vals := [op(K17vals), i]:
      od:
  fi:

  if type(K18R, integer) then
      K18vals:= [K18R]:
  else
      K18vals := []:
      for i from K18R[1] by K18R[3] to K18R[2] do
        K18vals := [op(K18vals), i]:
      od:
  fi:

  if type(K19R, integer) then
      K19vals:= [K19R]:
  else
      K19vals := []:
```

```
    for i from K19R[1] by K19R[3] to K19R[2] do
      K19vals := [op(K19vals), i]:
    od:
fi:

generate_inner(K17vals, K18vals, K19vals, ansTab, TS, vars,
update):
return(0):

end proc:


generate_inner := proc(K17vals, K18vals, K19vals, ansTab, TS,
vars, update)
local K17,K18,K19, st,et, TSX,TSY,TSZ, i, N,var,Lrt,sys, zero,
zeroInd,posCell, numRoots, d, cad, rcp, rcpolys, rcinequal,
count, R:

R := RegularChains:-PolynomialRing(vars):

count := 0:

for K17 in K17vals do
for K18 in K18vals do
for K19 in K19vals do

    st := time():

    # Substitutions
    TSZ := subs(k17=K17, k18=K18, k19=K19, TS):

    # Lazy Real Triangularize and checks
    Lrt := LazyRealTriangularize(TSZ, R):
    if type(Lrt,list)<>true then
      error("Lrt output format changed (not list) - probably
something lazy.  At: ", [K17, K18, K19]):
    fi:
    N := nops( Lrt ):
    if N<>1 then
      error("Multiple solution components found - unexpected.
At: ", [17, K18, K19] ): fi:
    sys := Lrt[1]:
    rcpolys := map(X->lhs(X), select(X->op(0,X)=`=`, sys)):
    rcinequal := map(X->rhs(X), remove(X->op(0,X)=`=`, sys)):
    N := nops(rcpolys):
    if N<>nops(vars) then
      error("Solution component has unexpected number of
equations.  At: ", [K17, K18, K19] ):
    fi:
    if rcinequal<>[vars[-1]] then
      error("Solution component has unexpected inequality.  At:
", [K17, K18, K19] ):
    fi:
    for i from 1 to nops(vars)-1 do:
      d := degree(rcpolys[i], vars[i]):
      if d <>1 then
        error("polynomials are not linear - unexpected.  At: ",
```

```
[K17, K18, K19] ):
      fi:
    od:
    d := degree(rcpolys[nops(vars)], vars[-1]):
    if d <>6 then
      error("Final polynomial is not degree 6 - unexpected.
At: ", [K17, K18, K19] ):
    fi:

    # Creating CAD and counting
    cad := CylindricalAlgebraicDecompose( [rcpolys[-1], vars
[-1]], PolynomialRing([vars[-1]]), output=list ):
    N := nops(cad):
    zero := select(X->X[2][2][1]=[0,0], cad);
    zeroInd := op(zero[1][1]);
    posCell := N-zeroInd;
    numRoots := (posCell-1)/2;

    if update = false then
      ;
    else
      count := count + 1:
      if modp(count,update)=0 then
          print("so far...", count):
      fi:
    fi:

    et := time() - st:
    ansTab[K17, K18, K19] := [numRoots, et]:


od: od: od:

end proc:


generate_howMany := proc(K17R, K18R, K19R, ansTab)
local check, r, K17vals, K18vals, K19vals, ans, i, count:

for r in [K17R, K18R, K19R] do
  check := (type(r, list) and nops(r)=3 and r[2]>r[1]) or type
(r, integer):
  if check<>true then
    error("First 3 arguments must each be an integer or a list
of 3 integers"):
  fi:
od:

if type(K17R, integer) then
    K17vals:= [K17R]:
else
    K17vals := []:
    for i from K17R[1] by K17R[3] to K17R[2] do
      K17vals := [op(K17vals), i]:
    od:
fi:
```

```
    if type(K18R, integer) then
        K18vals:= [K18R]:
    else
        K18vals := []:
        for i from K18R[1] by K18R[3] to K18R[2] do
          K18vals := [op(K18vals), i]:
        od:
    fi:

    if type(K19R, integer) then
        K19vals:= [K19R]:
    else
        K19vals := []:
        for i from K19R[1] by K19R[3] to K19R[2] do
          K19vals := [op(K19vals), i]:
        od:
    fi:

    print([nops(K17vals),nops(K18vals),nops(K19vals)]);
    return( nops(K17vals)*nops(K18vals)*nops(K19vals) ):

    end proc:
```

The problem = MAPK 026

Bio System of interest.

```
> TS := [
  k2*x6 - k1*x1*x4 - k16*x1*x5 + k15*x11 = 0,
  k3*x6 + k5*x7 + k10*x9 + k13*x10 - k11*x2*x5 - k12*x2*x5 - k4*
  x2*x4 = 0,
  k6*x7+k8*x8 - k7*x3*x5 = 0,
  -k1*x1*x4-k4*x2*x4+k2*x6+k3*x6+k5*x7+k6*x7 = 0,
  -k11*x2*x5-k12*x2*x5-k16*x1*x5-k7*x3*x5+k10*x9+k13*x10+k15*x11+
  k8*x8 = 0,
  k1*x1*x4-k2*x6-k3*x6 = 0,
  k4*x2*x4-k5*x7-k6*x7 = 0,
  k7*x3*x5-k8*x8-k9*x8 = 0,
  k11*x2*x5-k10*x9+k9*x8 = 0,
  k12*x2*x5-k13*x10-k14*x10 = 0,
  k16*x1*x5+k14*x10-k15*x11 = 0,
  -k17+x10+x11+x5+x8+x9 = 0,
  -k18+x4+x6+x7 = 0,
  -k19+x1+x10+x11+x2+x3+x6+x7+x8+x9 = 0
  ];
```

$$TS := [-k1\,x1\,x4 - k16\,x1\,x5 + k15\,x11 + k2\,x6 = 0,\ -k11\,x2\,x5 - k12\,x2\,x5 - k4\,x2\,x4 + k10\,x9 + k13\,x10 + k3\,x6 + k5\,x7 = 0,\ -k7\,x3\,x5 + k6\,x7 + k8\,x8 = 0,\ -k1\,x1\,x4 - k4\,x2\,x4 + k2\,x6 + k3\,x6 + k5\,x7 + k6\,x7 = 0,\ -k11\,x2\,x5 - k12\,x2\,x5 - k16\,x1\,x5 - k7\,x3\,x5 + k10\,x9 + k13\,x10 + k15\,x11 + k8\,x8 = 0,\ k1\,x1\,x4 - k2\,x6 - k3\,x6 = 0,\ k4\,x2\,x4 - k5\,x7 - k6\,x7 = 0,\ k7\,x3\,x5 - k8\,x8 - k9\,x8 = 0,\ k11\,x2\,x5 - k10\,x9 + k9\,x8 = 0,\ k12\,x2\,x5 - k13\,x10 - k14\,x10 = 0,\ k16\,x1\,x5 + k14\,x10 - k15\,x11 = 0,\ -k17 + x10 + x11 + x5 + x8 + x9 = 0,\ -k18 + x4 + x6 + x7 = 0,\ -k19 + x1 + x10 + x11 + x2 + x3 + x6 + x7 + x8 + x9 = 0]$$

9 of the parameters have been measured accuretley.

```
> s1 := {k1 = 0.2e-1, k11 = 0.1e-1, k12 = 0.1e-1, k15 = 0.86e-1,
  k16 = 0.11e-2, k3 = 0.1e-1, k4 = 0.32e-1, k7 = 0.45e-1, k9 =
  0.92e-1};
  nops(%);
```

$$s1 := \{k1 = 0.02, k11 = 0.01, k12 = 0.01, k15 = 0.086, k16 = 0.0011, k3 = 0.01, k4 = 0.032, k7 = 0.045, k9 = 0.092\}$$

9

Another 7 have been estimated with confidence

```
> s2 := {k10 = 1, k13 = 1, k14 = .5, k2 = 1, k5 = 1, k6 = 15, k8
  = 1};
  nops(%);
```

$$s2 := \{k10 = 1, k13 = 1, k14 = 0.5, k2 = 1, k5 = 1, k6 = 15, k8 = 1\}$$

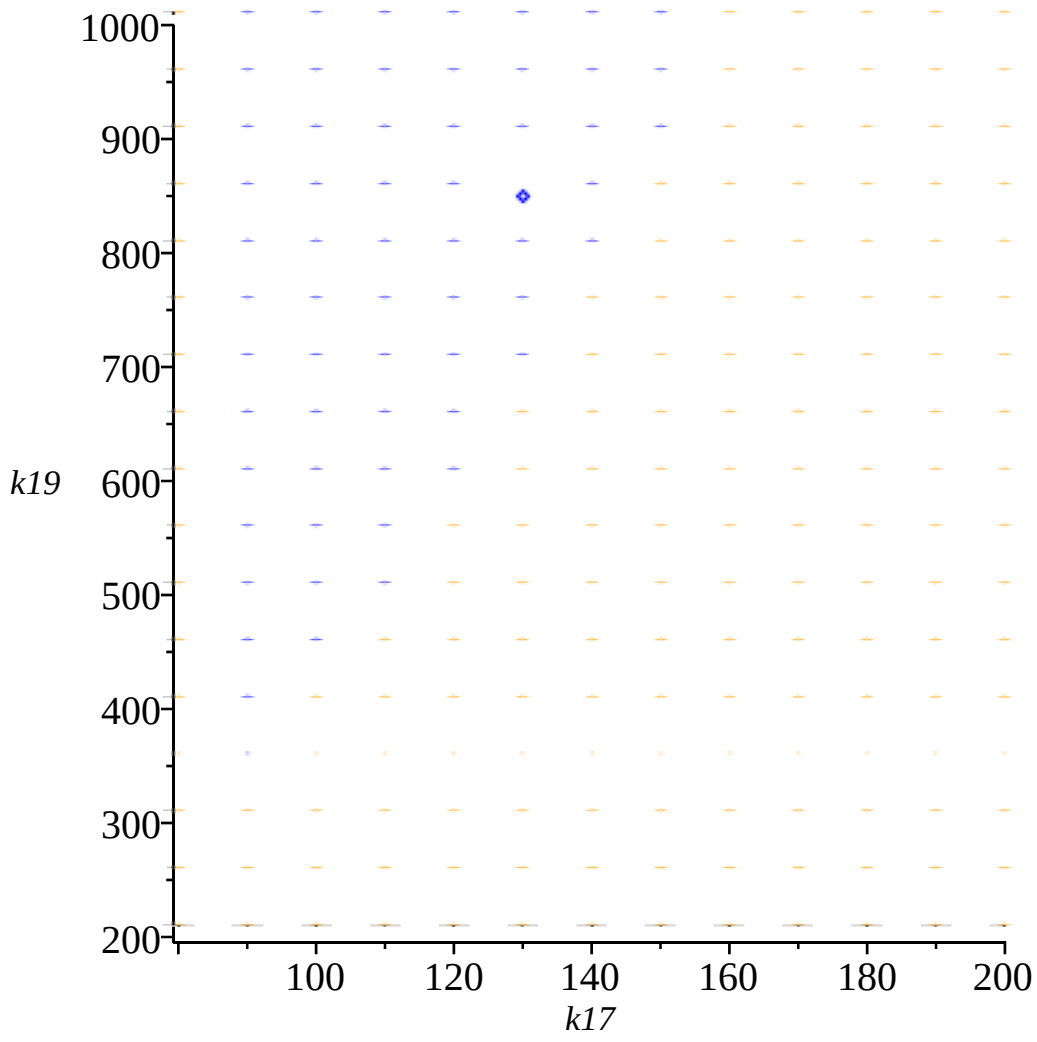
7

Graphs and Timings

```
> answers := table():
  generate([80,200,10], 50, [200,1000,50], answers, TSX, vars ):
> save(answers, "SamplePoints-Sys26-Original-Graph1.txt")
> read("SamplePoints-Sys26-Original-Graph1.txt"):
> inds := [indices( answers )]: nops(%);
  ents := [entries( answers )]: nops(%):
  nums := map(X->X[1][1], ents):
  cols := []:
  for ent in nums do:
  if ent=1 then
    cols := [op(cols), "Orange"]:
  else:
    cols := [op(cols), "Blue"]:
  fi:
  od:
```

221

```
> plots :- pointplot( map(X->[X[1],X[3]],inds), color = cols,
  labels=[k17, k19] );
```

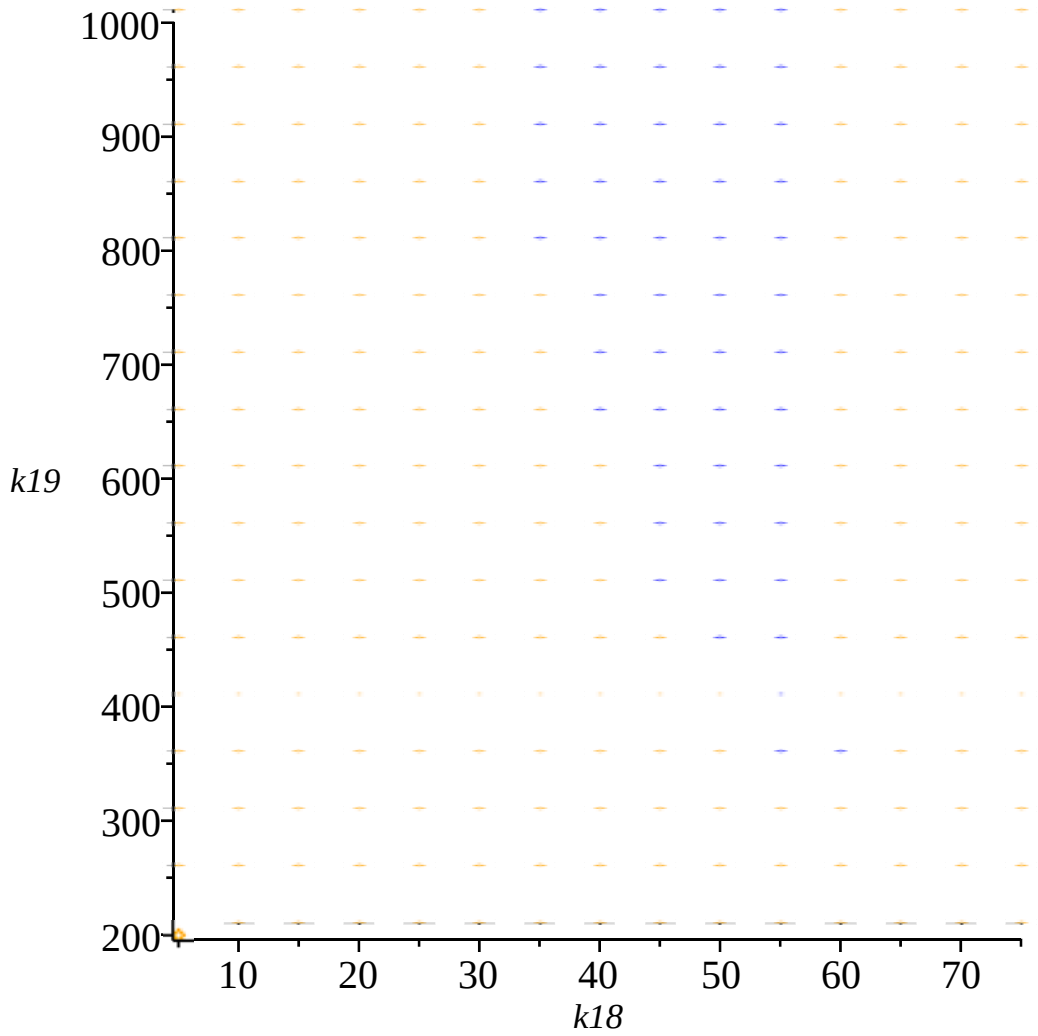


```
> timeLorg:=map(X->X[1][2],ents):

> answers := table():
  generate(100, [5,75,5], [200,1000,50], answers, TSX, vars ):
> save(answers, "SamplePoints-Sys26-Original-Graph2.txt")
> read("SamplePoints-Sys26-Original-Graph2.txt"):
> inds := [indices( answers )]: nops(%);
  ents := [entries( answers )]: nops(%):
  nums := map(X->X[1][1], ents):
  cols := []:
  for ent in nums do:
  if ent=1 then
    cols := [op(cols), "Orange"]:
  else:
    cols := [op(cols), "Blue"]:
  fi:
  od:
```

255

```
> plots :- pointplot( map(X->[X[2],X[3]],inds), color = cols,
  labels=[k18, k19] );
```




```
> timeRorg:=map(X->X[1][2],ents):
```

```
> timeOrg := [op(timeLorg), op(timeRorg)]: nops(%);
```

510

Averages

```
> add(timeLorg)/nops(timeLorg), add(timeRorg)/nops(timeRorg), add
  (timeOrg)/nops(timeOrg);
```

0.5680274510, 0.5680274510, 0.5680274510

Maximum

```
> max(timeLorg), max(timeRorg);
```

0.905, 0.905

Minimum

```
> min(timeLorg), min(timeRorg);
```

0.390, 0.390

Median

```
> Statistics:-Median(timeLorg), Statistics:-Median(timeRorg),
  Statistics:-Median(timeOrg);
```

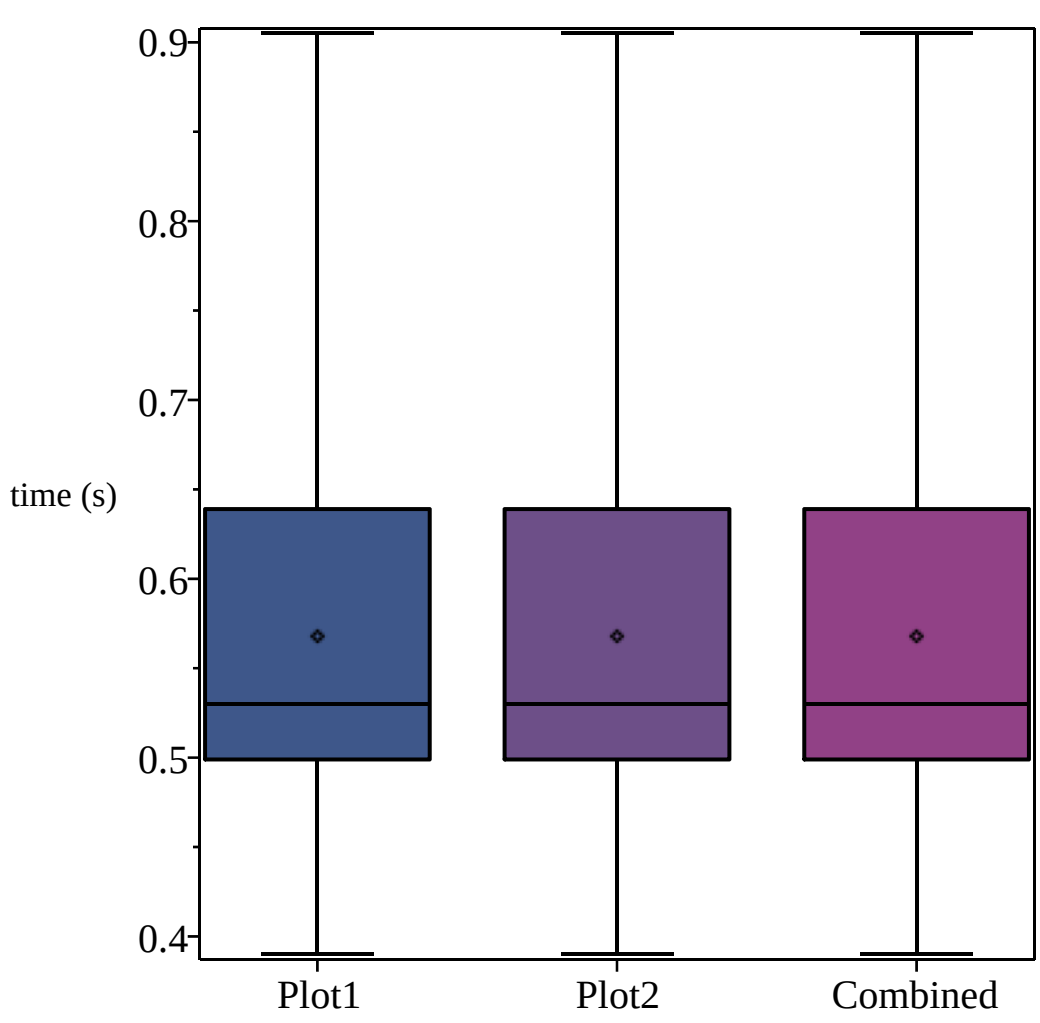
0.530000000000000, 0.530000000000000, 0.530000000000000

SD

```
> Statistics:-StandardDeviation(timeLorg), Statistics:-
  StandardDeviation(timeRorg), Statistics:-StandardDeviation
  (timeOrg);
```

0.106795720958896, 0.106795720958896, 0.106690761994022

```
> Statistics:-BoxPlot( [timeLorg, timeRorg, timeOrg], mean=true,
  deciles=false, outliers=false, datasetlabels = ["Plot1",
  "Plot2", "Combined"], labels=["", "time (s)"] );
```


0.9
0.8
0.7
0.6
0.5
0.4
time (s)
Plot1
Plot2
Combined

Reduced Problem

```
> TS := [
    1062444*k18*x4**2*x5 + 23478000*k18*x4**2 + 1153450*k18*x4*
  x5**2 + 2967000*k18*x4*x5 + 638825*k18*x5**3 + 49944500*k18*x5*
  *2 - 5934*k19*x4**2*x5 - 989000*k19*x4*x5**2 - 1062444*x4**3*x5
  - 23478000*x4**3 - 1153450*x4**2*x5**2 - 2967000*x4**2*x5 -
  638825*x4*x5**3 - 49944500*x4*x5**2 = 0,
    1062444*k17*x4**2*x5 + 23478000*k17*x4**2 + 1153450*k17*x4*
  x5**2 + 2967000*k17*x4*x5 + 638825*k17*x5**3 + 49944500*k17*x5*
  *2 - 1056510*k19*x4**2*x5 - 164450*k19*x4*x5**2 - 638825*k19*
  x5**3 - 1062444*x4**2*x5**2 - 23478000*x4**2*x5 - 1153450*x4*
  x5**3 - 2967000*x4*x5**2 - 638825*x5**4 - 49944500*x5**3 = 0
  ];
```

$$TS := [1062444\,k18\,x4^2\,x5 + 1153450\,k18\,x4\,x5^2 + 638825\,k18\,x5^3 - 5934\,k19\,x4^2\,x5 - 989000\,k19\,x4\,x5^2 - 1062444\,x4^3\,x5 - 1153450\,x4^2\,x5^2 - 638825\,x4\,x5^3 + 23478000\,k18\,x4^2 + 2967000\,k18\,x4\,x5 + 49944500\,k18\,x5^2 - 23478000\,x4^3 - 2967000\,x4^2\,x5 - 49944500\,x4\,x5^2 = 0,\ 1062444\,k17\,x4^2\,x5 + 1153450\,k17\,x4\,x5^2 + 638825\,k17\,x5^3 - 1056510\,k19\,x4^2\,x5 - 164450\,k19\,x4\,x5^2 - 638825\,k19\,x5^3 - 1062444\,x4^2\,x5^2 - 1153450\,x4\,x5^3 - 638825\,x5^4 + 23478000\,k17\,x4^2 + 2967000\,k17\,x4\,x5 + 49944500\,k17\,x5^2 - 23478000\,x4^2\,x5 - 2967000\,x4\,x5^2 - 49944500\,x5^3 = 0]$$

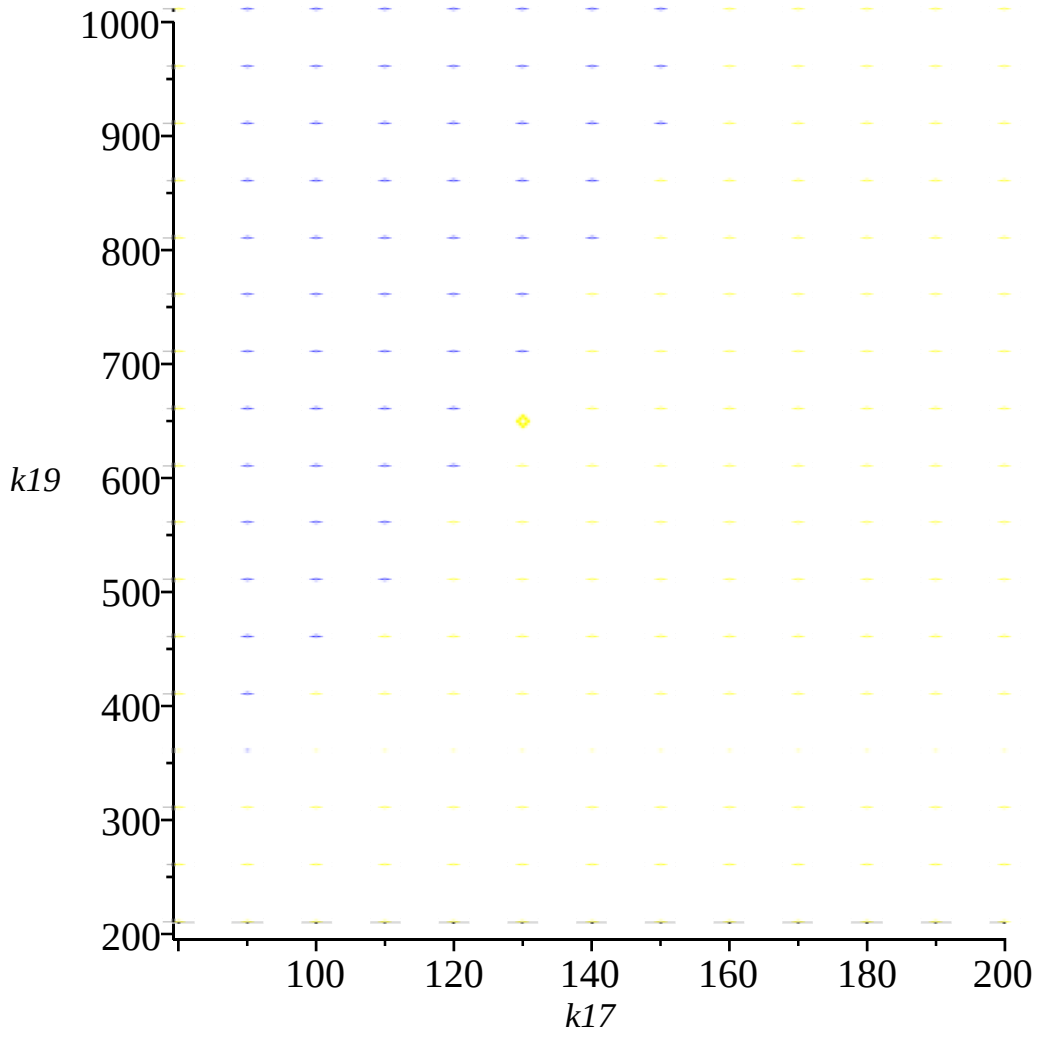
```
> TSX := [op(TS), x4>0, x5>0]:
> vars := [x4,x5]: R:=PolynomialRing(vars):
```

Graphs and Timings

```
> answers := table():
  generate([80,200,10], 50, [200,1000,50], answers, TSX, vars ):
> save(answers, "SamplePoints-Sys26-Reduced-Graph1.txt")
> read("SamplePoints-Sys26-Reduced-Graph1.txt"):
> inds := [indices( answers )]: nops(%);
  ents := [entries( answers )]: nops(%):
  nums := map(X->X[1][1], ents):
  cols := []:
  for ent in nums do:
  if ent=1 then
    cols := [op(cols), "Yellow"]:
  else:
    cols := [op(cols), "Blue"]:
  fi:
  od:
```

221

```
> plots :- pointplot( map(X->[X[1],X[3]],inds), color = cols,
  labels=[k17, k19] );
```

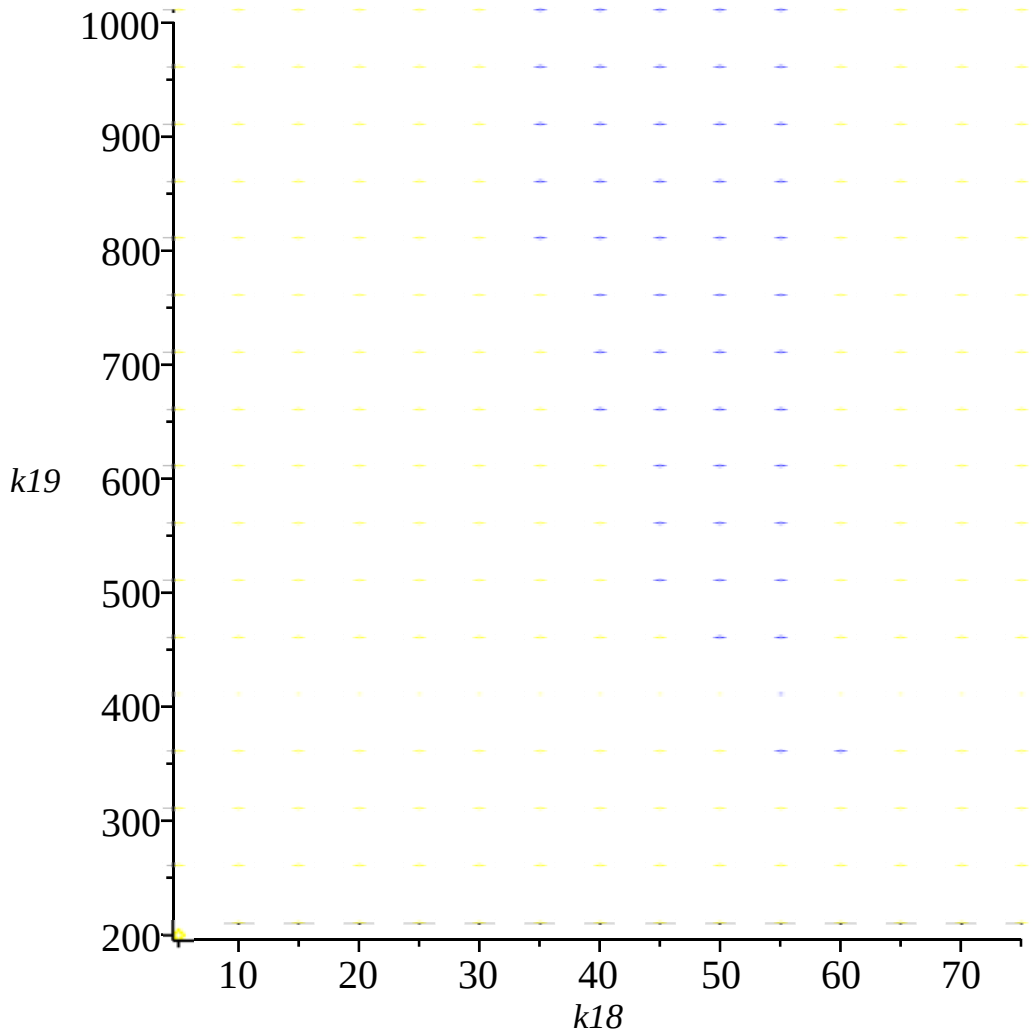


```
> timeLred:=map(X->X[1][2],ents):

> answers := table():
  generate(100, [5,75,5], [200,1000,50], answers, TSX, vars ):
> save(answers, "SamplePoints-Sys26-Reduced-Graph2.txt")
> read("SamplePoints-Sys26-Reduced-Graph2.txt"):
> inds := [indices( answers )]: nops(%);
  ents := [entries( answers )]: nops(%):
  nums := map(X->X[1][1], ents):
  cols := []:
  for ent in nums do:
  if ent=1 then
    cols := [op(cols), "Yellow"]:
  else:
    cols := [op(cols), "Blue"]:
  fi:
  od:
```

255

```
> plots :- pointplot( map(X->[X[2],X[3]],inds), color = cols,
  labels=[k18, k19] );
```




```
> timeRred:=map(X->X[1][2],ents):
```

```
> timeRed := [op(timeLred), op(timeRred)]: nops(%);
                                    476
```

Averages

```
> add(timeLred)/nops(timeLred), add(timeRred)/nops(timeRred), add
  (timeRed)/nops(timeRed);
          0.05322624434, 0.05254901961, 0.05286344538
> av := add(timeRed)/nops(timeRed):
```

Maximum

```
> max(timeLred), max(timeRred);
                                  0.327, 0.343
```

Minimum

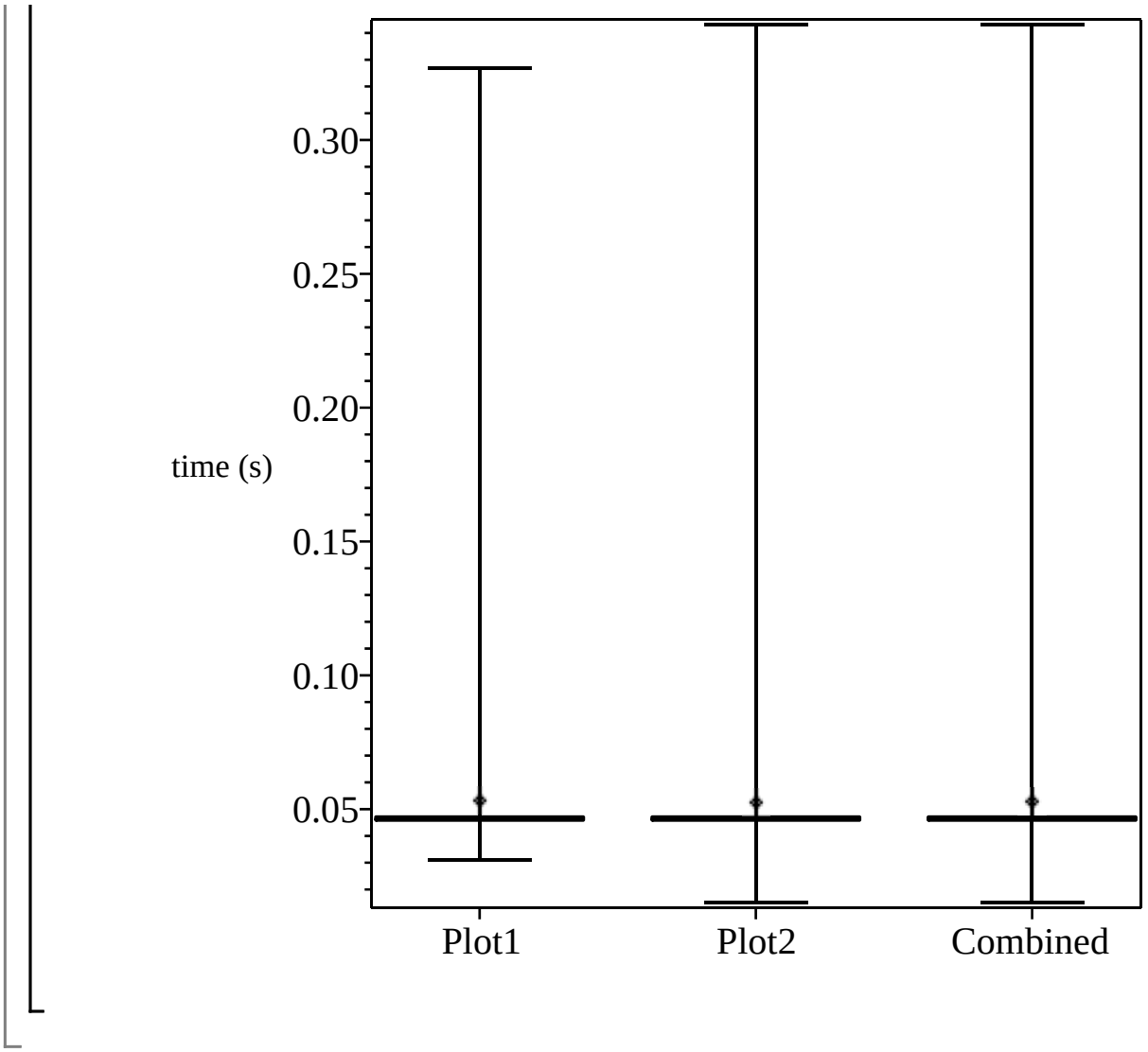
```
> min(timeLred), min(timeRred);
                                  0.031, 0.015
```

Median

```
> Statistics:-Median(timeLred), Statistics:-Median(timeRred),
  Statistics:-Median(timeRed);
     0.0470000000000000, 0.0470000000000000, 0.0470000000000000
```

SD

```
> Statistics:-StandardDeviation(timeLred), Statistics:-
  StandardDeviation(timeRred), Statistics:-StandardDeviation
  (timeRed);
     0.0362326251641456, 0.0367296538168751, 0.0364629335641605
> Statistics:-BoxPlot( [timeLred, timeRred, timeRed], mean=true,
  deciles=false, outliers=false, datasetlabels = ["Plot1",
  "Plot2", "Combined"], labels=["", "time (s)"] );
```



▶ Final High Res Graphs - Sys26 LEFT

▶ Final High Res Graphs - Sys26 RIGHT