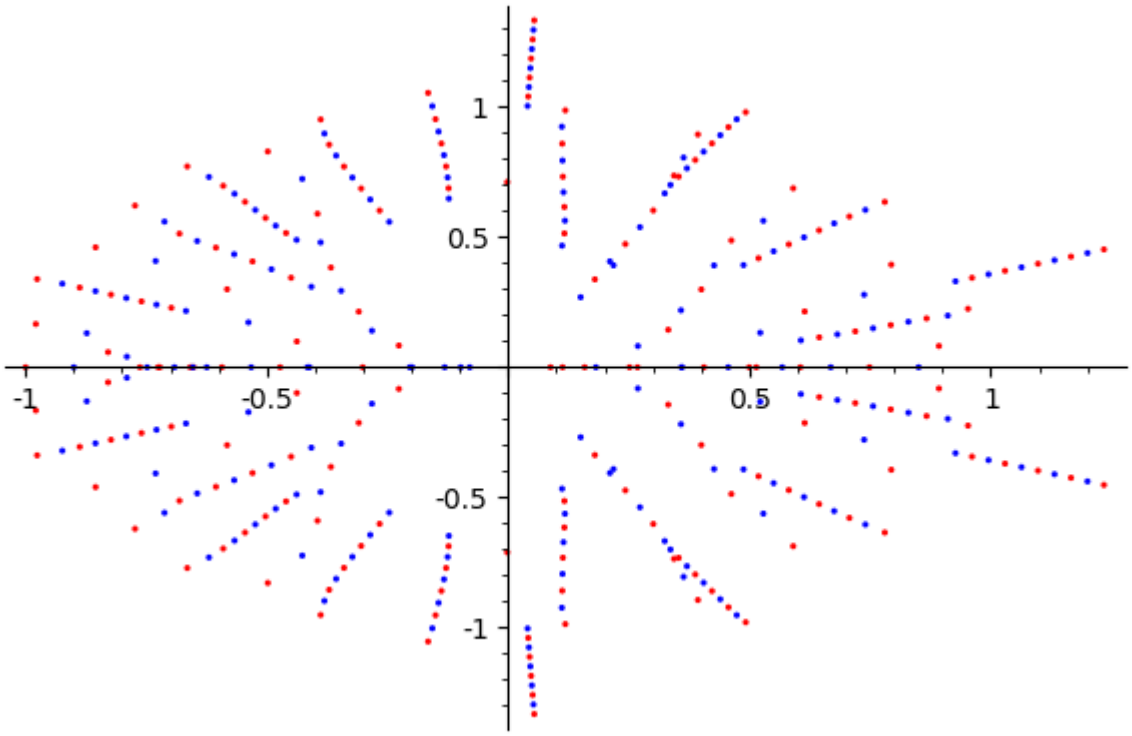
In [172]:
```
#  Problem 1

reset()
def eulRec(n):
    var('t')
    if n == 0:
        return 1
    F(t) = eulRec(n-1)
    diffN_1 = diff(F,t)
    result = t*(1-t)*diffN_1 + eulRec(n-1) * (1+(n-1))*t
    return result

def mEulRec(n):
    var('t')
    resultList = [1]  # n is 0 return 1
    for i in range(1,n+1):
        F(t) = resultList[-1]
        temp = t*(1-t)* (F.diff(t)) + resultList[-1] * (1+(i-1))*t
        resultList.append(temp)
    return resultList[n]

def eulDirect(n):
    var('t')
    return  (t+1)^n + (t-1)^(n-1) - (n-1)
#     return (1-t)^(n-1) + t * (1+n)^n

def eulRoots(n):
    return mEulRec(10).roots(ring=CC,multiplicities=False)

print(eulRec(4).expand())
print(mEulRec(5).expand())
print(eulDirect(6).expand())
print(eulRoots(10))
```

```
t |--> t^4 + 11*t^3 + 11*t^2 + t
t |--> t^5 + 26*t^4 + 66*t^3 + 26*t^2 + t
t^6 + 7*t^5 + 10*t^4 + 30*t^3 + 5*t^2 + 11*t - 5
[(-1, 1), (0, 1)]
```

In [157]:
```
# Problem 2

reset()
import random

def problem2(d,j):
    randList = [-1,1]
    f=0
    for i in range(d+1):
        r = random.choice(randList)
        f = f + r*x^i
    show(f)

    oddP, evenP = [],[]
    for i in range(j):
        df = diff(f,x,i)
#         print(df.roots(ring = CC))
        for a in df.roots(ring = CC, multiplicities=False):
            if i%2 == 0:  #even
#                 evenP.append((real_part(a[0]), real_part(a[1])))
                evenP.append(a)
            else:
#                 oddP.append((real_part(a[0]), real_part(a[1])))
                oddP.append(a)
#     print(evenP)
#     print(oddP)
    evenPt = point(evenP, color='red', size = 5)
    oddPt = point(oddP, color='blue', size = 5)
    plot(evenPt+oddPt).show()

problem2(40,10)
```

$$-x^{39} + x^{38} - x^{37} + x^{36} + x^{35} - x^{34} - x^{33} - x^{32} - x^{31} - x^{30} - x^{29} + x^{28} + x^{27} + x^{2}$$
$$- x^{17} - x^{16} + x^{15} + x^{14} - x^{13} + x^{12} + x^{11} - x^{10} - x^{9} + x^{8} + x^{7} -$$



In [166]:
```
# Problem 3

reset()
var ('x,y')
def problem3(f):
    fxx = diff(f,x,2)
    fyy = diff(f,y,2)
    fxy = diff(diff(f,x),y)
    d = fxx*fyy-fxy^2
    criticalP = solve([f.diff(x)==0,f.diff(y)==0],[x,y],solution_dict=True)

    resultList = []
    for i in range(len(criticalP)):
        if d.subs(criticalP[i]) > 0 and fxx.subs(criticalP[i]) > 0:
            resultList.append([(criticalP[i][x],criticalP[i][y]),'local min'])
        elif d.subs(criticalP[i]) > 0 and fxx.subs(criticalP[i]) < 0:
            resultList.append([(criticalP[i][x],criticalP[i][y]),'local max'])
        elif d.subs(criticalP[i]) < 0:
            resultList.append([(criticalP[i][x],criticalP[i][y]),'saddle poin
t'])
        elif d.subs(criticalP[i]) == 0:
            resultList.append([(criticalP[i][x],criticalP[i][y]),'inconclusiv
e'])

    return resultList

# print(problem3(y^3 + 3*x^2*y - 6*x^2 - 6*y^2 + 2))
problem3(3*y^3 - x^2*y^2 + 8*y^2+4*x^2-20*y)
```

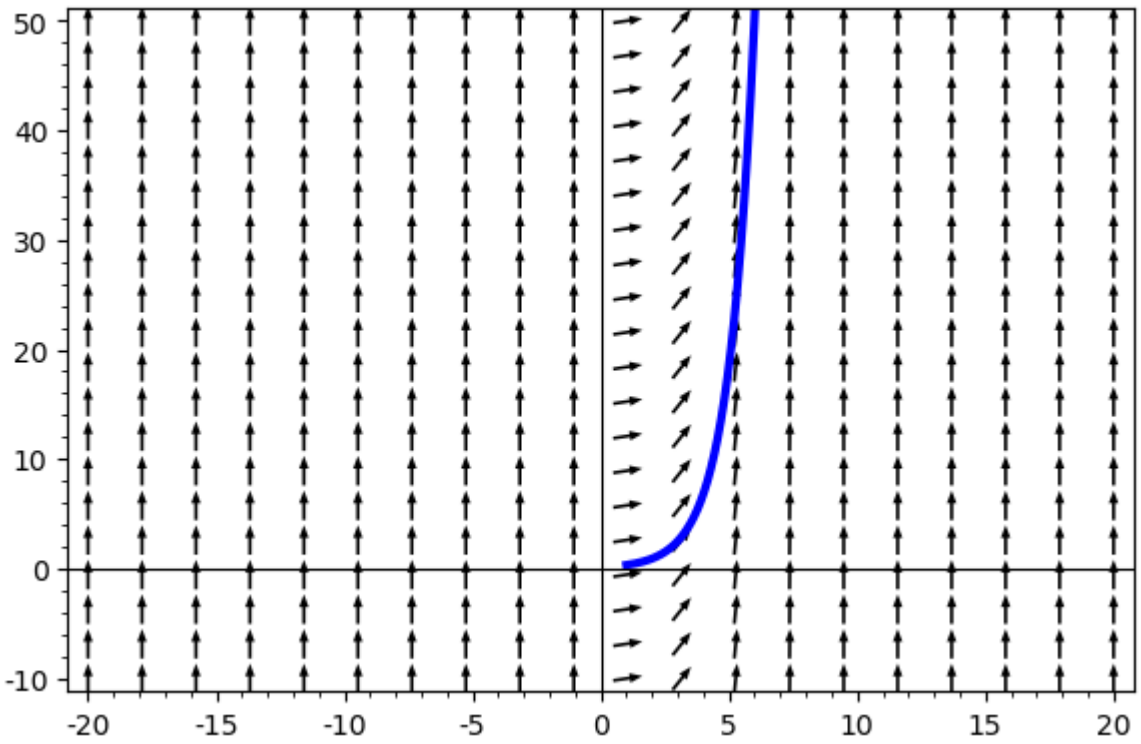
Out[166]:
```
[[(0, -2/9*sqrt(61) - 8/9), 'local max'],
 [(0, 2/9*sqrt(61) - 8/9), 'local min'],
 [(2, -2), 'saddle point'],
 [(-2, -2), 'saddle point'],
 [(-2*sqrt(3), 2), 'saddle point'],
 [(2*sqrt(3), 2), 'saddle point']]
```

In [159]:
```
# Problem 4

reset()
def problem4():
    var('t')
    y = function('y')(t)
    myDiffEq = diff(y, t) + 7*y == exp(t)

    s = desolve(myDiffEq, y)
    show(s)

    s2 = desolve(myDiffEq, y, [0,0])
    show(s2)

    der = solve(myDiffEq,diff(y,t))
    d = der[0].rhs()
    d2 = d.subs({y(t):s2 })
    A = plot_slope_field(d2,(t,-20,20),(y,-10,50),headlength=4,headaxislength=
4)
    B = plot(s2, (t,1,20),ymin =-10,ymax = 50,thickness=3)
    (A+B).show()

problem4()
```

$$\frac{1}{8}\left(8\,C + e^{(8\,t)}\right)e^{(-7\,t)}$$

$$\frac{1}{8}\left(e^{(8\,t)} - 1\right)e^{(-7\,t)}$$

```
/opt/sagemath-9.1/local/lib/python3.7/site-packages/sage/all_cmdline.py:22: D
eprecationWarning: Substitution using function-call syntax and unnamed argume
nts is deprecated and will be removed from a future release of Sage; you can
use named arguments instead, like EXPR(x=..., y=...)
See http://trac.sagemath.org/5930 for details.
```



In [160]:
```
# Problem 5

reset()
def expApprox(M,n):
    a = M^0 / factorial(0)
    for i in range(1,n):
        a += M^i / factorial(i)
    return a

def problem5():
    v = expApprox(M,10).eigenvalues()
    sorted(v)
    show(v)

    wList = []
    for i in range(len(v)):
        wList.append(exp(v[i]))
    w = vector(wList)
    show(w)

    B = expApprox(M,50)
#     show(B)

    u = B.eigenvalues()
    sorted(u)
    uV = vector([u[0],u[1],u[2],u[3]])
    show(uV)

#     d = abs(uV-w)
    show(norm(uV-w))

M = matrix([[1,2,3,4],[2,1,5,7],[3,5,1,8],[4,7,8,9]])
show(expApprox(M,10))
problem5()
```

$$\begin{pmatrix} \frac{14963472679}{90720} & \frac{9250807591}{36288} & \frac{20336879935}{72576} & \frac{158115854471}{362880} \\ \frac{9250807591}{36288} & \frac{71487836951}{181440} & \frac{19644929111}{45360} & \frac{11637037079}{17280} \\ \frac{20336879935}{72576} & \frac{19644929111}{45360} & \frac{9597002399}{20160} & \frac{9593499037}{12960} \\ \frac{158115854471}{362880} & \frac{11637037079}{17280} & \frac{9593499037}{12960} & \frac{52211395343}{45360} \end{pmatrix}$$

$$[-5.711009524351727?, -3.977375443777415?, -0.7984506828163931?, 2.186041049$$

$$\left(e^{(-5.711009524351727?)}, e^{(-3.977375443777415?)}, e^{(-0.7984506828163931?)}, e^{2.186041049401789?e6}\right)$$

$$(-5.286142046415604?, -3.964082672174489?, -0.7984506827418292?, 3.7952986$$

$$\sqrt{\left(e^{2.186041049401789?e6} - 3.795298668566275?e8\right)^2 + \left(e^{(-0.7984506828163931?)} + 0.7984\right.}$$
$$\left. + \left(e^{(-3.977375443777415?)} + 3.964082672174489?\right)^2 + \left(e^{(-5.711009524351727?)} + 5.286\right.$$