Stability of a fixed point in a system of ODE

Yasuyuki Nakamura

Graduate School of Information Science, Nagoya University

A4-2(780), Furo-cho, Chikusa-ku, Nagoya, 464-8601, Japan

nakamura@nagoya-u.jp

http://www.phys.cs.is.nagoya-u.ac.jp/~nakamura/

Let us consider the following system of ODE

$$\frac{dx}{dt} = f_1(x, y, z, \ldots)$$
$$\frac{dy}{dt} = f_2(x, y, z, \ldots)$$
$$\frac{dz}{dt} = f_3(x, y, z, \ldots)$$
$$\ldots$$

System of ODE

For the simplisity, we consider the follwoing system of autonomous ODE with two variables.
(Please input $x(t)$ and $y(t)$ without independent variable t , like x for $x(t)$ and y for $y(t)$.)

$$\frac{dx}{dt} = \texttt{-y+y\^{}3} \equiv f_1(x, y)$$

$$\frac{dy}{dt} = \texttt{sin(x)-y/2} \equiv f_2(x, y)$$

Fixed point

Fixed points (x, y) are defined with the condition $f_1(x, y) = 0,\ f_2(x, y) = 0$. Let one of them to be (x_0, y_0). Note that there could be more than one fixed points.

Calc fixed point

$(x_0, y_0) =$ (0, 0), (1/6*Pi, 1), (-1/6*Pi, -1)

(Note, when solutions are not expressed in explicit form, the solution are not listed above.)

Linearization

In order to analize a behaviour of solutions near fixed points, let us consider the system of ODE for $X \equiv x - x_0$, $Y \equiv y - y_0$. We linearize the original ODE under the condition $X, Y \ll 1$.

When we linearize ODE near 1 th fixed point $(x_0, y_0) =$ (0 , 0), ODE for X, Y is calculated to be as follows.

Linearization

$\frac{dX}{dt} =$ $-Y$

$\frac{dY}{dt} =$ $X - \frac{1}{2} Y$

in matrix form, $\frac{d}{dt}\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & -\frac{1}{2} \end{bmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} \equiv A \begin{pmatrix} X \\ Y \end{pmatrix}$

Stability of fixed points

Stability of a fixed point can be determined by eigen values of matrix A.

Eigen values of A are

$-\frac{1}{4} + \frac{1}{4} I\sqrt{15}$,

$-\frac{1}{4} - \frac{1}{4} I\sqrt{15}$, therfore

the fixed point (0 , 0) is

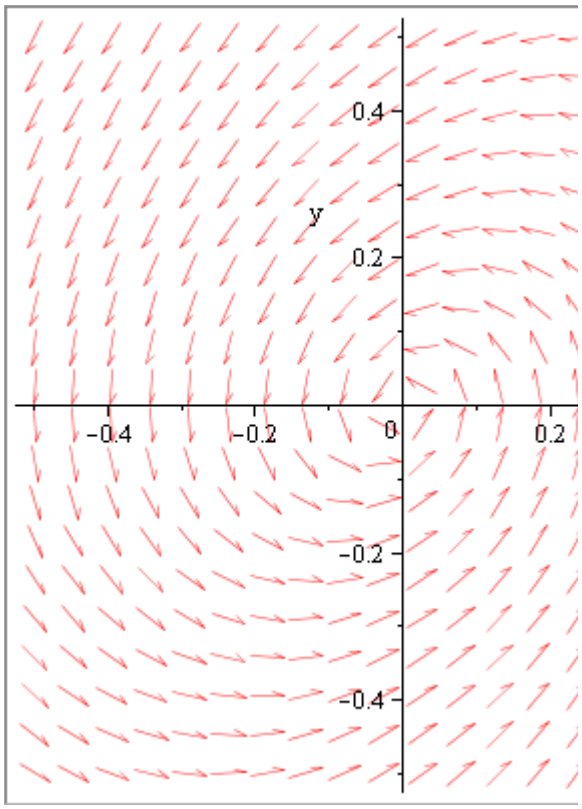
asymptotic .

Direction field near the fixed point (0 , 0) is displayed in the right figure.

Calculation

Solution curve starting (0.4 , 0) can also diplayed with animation.

Solution curve ($0 \le t \le$ 20)



Function

```
> restart:
with(DocumentTools):
with(LinearAlgebra):
with(DEtools):
with(plots):
with(StringTools):
> odes := proc()
  global ode1, ode2;
```

```
  ode1 := parse(GetProperty(t_ode1, value));
  ode2 := parse(GetProperty(t_ode2, value));
end proc:
> sol := proc()
  global ode1, ode2, fp, s_fp;

#  odes();
  ode1 := parse(GetProperty(t_ode1, value));
  ode2 := parse(GetProperty(t_ode2, value));
  fp:=solve({ode1=0., ode2=0.}, [x, y]);
  SetProperty(t_fp, value, st_sol());
  SetProperty(t_no_fp, value, 1);
  show_sol();
#  SetProperty(fp_x, value, NULL);
#  SetProperty(fp_y, value, NULL);
#  SetProperty(fp_x2, value, NULL);
#  SetProperty(fp_y2, value, NULL);
#  SetProperty(fp_x3, value, NULL);
#  SetProperty(fp_y3, value, NULL);
  SetProperty(m_ode1_l, value, NULL);
  SetProperty(m_ode2_l, value, NULL);
  SetProperty(m_matrix, value, NULL);
#  SetProperty(eigen_1, value, NULL);
#  SetProperty(eigen_2, value, NULL);
  SetProperty(t_ix, value, NULL);
  SetProperty(t_iy, value, NULL);
  SetProperty(stability, value, NULL);
end proc:
> st_sol := proc()
  global fp;
  local i, s_fp;

  s_fp := "";
  for i from 1 to nops(fp) do
     if(type(rhs(fp[i][1]), imaginary) or
type(rhs(fp[i][2]), imaginary)) then next; end if;
     s_fp := cat(s_fp, "(",
convert(identify(rhs(fp[i][1])), string), ", ",
convert(identify(rhs(fp[i][2])), string), "), ");
  end do;
  if(s_fp = "") then
    "There is no real solution.";
  else
    substring(s_fp, 1..-3);
  end if;
end proc:
> show_sol := proc()
```

```
  global fp;
  local i;

  if(GetProperty(t_fp, value)="There is no real solution.")
then
    SetProperty(t_no_fp, value, NULL);
    SetProperty(fp_x, value, NULL);
    SetProperty(fp_y, value, NULL);
    SetProperty(fp_x2, value, NULL);
    SetProperty(fp_y2, value, NULL);
    SetProperty(fp_x3, value, NULL);
    SetProperty(fp_y3, value, NULL);
  else
    i := parse(GetProperty(t_no_fp, value));
    SetProperty(fp_x, value, identify(rhs(fp[i][1])));
    SetProperty(fp_y, value, identify(rhs(fp[i][2])));
    SetProperty(fp_x2, value, identify(rhs(fp[i][1])));
    SetProperty(fp_y2, value, identify(rhs(fp[i][2])));
    SetProperty(fp_x3, value, identify(rhs(fp[i][1])));
    SetProperty(fp_y3, value, identify(rhs(fp[i][2])));
  end if;
  SetProperty(eigen_1, value, NULL);
  SetProperty(eigen_2, value, NULL);
end proc:
> lin := proc()
  global A, l1, l2;
  local i;

  i := parse(GetProperty(t_no_fp, value));
  l1 := mtaylor(ode1, [x=identify(rhs(fp[i][1])),
y=identify(rhs(fp[i][2]))], 2);
  l2 := mtaylor(ode2, [x=identify(rhs(fp[i][1])),
y=identify(rhs(fp[i][2]))], 2);
  l1 := subs(x=X+identify(rhs(fp[i][1])),
y=Y+identify(rhs(fp[i][2])), l1);
  l2 := subs(x=X+identify(rhs(fp[i][1])),
y=Y+identify(rhs(fp[i][2])), l2);
  SetProperty(m_ode1_l, value, l1);
  SetProperty(m_ode2_l, value, l2);

  A := Matrix([[coeff(l1, X), coeff(l1, Y)], [coeff(l2, X),
coeff(l2, Y)]]);
  SetProperty(m_matrix, value, A);

  SetProperty(eigen_1, value, NULL);
  SetProperty(eigen_2, value, NULL);
  SetProperty(stability, value, NULL);
```

```
  SetProperty(Plot0, value, NULL);
end proc:
> calc := proc()
  global A, ode1, ode2;
  local i, iv1, iv2, xmin, xmax, ymin, ymax, p, ode1_new,
ode2_new;

  iv1 := Eigenvalues(A)[1];
  iv2 := Eigenvalues(A)[2];
#  iv1 := identify(iv1);
#  iv2 := identify(iv2);
  SetProperty(eigen_1, value, iv1);
  SetProperty(eigen_2, value, iv2);

  if (Im(iv1)=0) then
    if (evalf(iv1)>0 and evalf(iv2)>0) then
SetProperty(stability, value, "unstable") end if;
    if (evalf(iv1)<0 and evalf(iv2)<0) then
SetProperty(stability, value, "asymptotic") end if;
    if (evalf(iv1)*evalf(iv2) < 0) then
SetProperty(stability, value, "saddle") end if;
  else
    if (Re(iv1)=0) then
      SetProperty(stability, value, "circle");
    elif (Re(iv1)>0) then
      SetProperty(stability, value, "unstable spiral");
    elif (Re(iv1)<0) then
      SetProperty(stability, value, "asymptotic stable
spiral");
    end if;
  end if;

  i := parse(GetProperty(t_no_fp, value));
  xmin := rhs(fp[i][1])-1/2;
  xmax := rhs(fp[i][1])+1/2;
  ymin := rhs(fp[i][2])-1/2;
  ymax := rhs(fp[i][2])+1/2;
  ode1_new :=
parse(SubstituteAll(SubstituteAll(convert(ode1, string),
"x", "x(t)"), "y", "y(t)"));
  ode2_new :=
parse(SubstituteAll(SubstituteAll(convert(ode2, string),
"x", "x(t)"), "y", "y(t)"));
  p := DEplot({diff(x(t), t)=ode1_new, diff(y(t),
t)=ode2_new}, [x(t), y(t)], t=0..1, x=xmin..xmax,
y=ymin..ymax);
  SetProperty(Plot0, value, display(p));
```

```
end proc:
> orbit := proc()
  global ode1, ode2;
  local i, iv1, iv2, xmin, xmax, ymin, ymax, p, ix, iy,
ode1_new, ode2_new, tmax;

  i := parse(GetProperty(t_no_fp, value));
  ix := parse(GetProperty(t_ix, value));
  iy := parse(GetProperty(t_iy, value));
  xmin := rhs(fp[i][1])-1/2;
  xmax := rhs(fp[i][1])+1/2;
  ymin := rhs(fp[i][2])-1/2;
  ymax := rhs(fp[i][2])+1/2;
  ode1_new :=
parse(SubstituteAll(SubstituteAll(convert(ode1, string),
"x", "x(t)"), "y", "y(t)"));
  ode2_new :=
parse(SubstituteAll(SubstituteAll(convert(ode2, string),
"x", "x(t)"), "y", "y(t)"));
  tmax := parse(GetProperty(t_time, value));
  p := DEplot({diff(x(t), t)=ode1_new, diff(y(t),
t)=ode2_new}, [x(t), y(t)], t=0..tmax, [[x(0)=ix,
y(0)=iy]], x=xmin..xmax, y=ymin..ymax, numpoints=5*tmax,
animatecurves=true, linecolor=red);
  SetProperty(Plot0, value, display(p, insequence=true));
  SetProperty(Plot0, play, true);
end proc:
>
```