Some solutions for Homework Set 2

```
> with(DEtools): with(plots):
```

Problem 2.6.1

While the solutions of the undamped spring-mass system

$$\frac{m\,d^2x}{dt^2} = -k\,x$$

indeed oscillate in one spatial dimension, this does not contradict the result that one-dimensional dynamical systems do not oscillate, since *as a dynamical system* this system is two-dimensional (it is equivalent to two coupled first-order differential equations).

Problem 2.7.6

```
> f1 := r + x - x^3;
  V1 := - int(f1,x);
```

$$f1 := r + x - x^3$$

$$V1 := -r\,x - \frac{1}{2}x^2 + \frac{1}{4}x^4$$

In order to plot all qualitatively different potential functions V, we need to find the bifurcation values:

```
solve({f1=0,diff(f1,x)=0},{x,r});
```

$$\{x = \text{RootOf}(-1 + 3\,\_Z^2,\ label = \_L3),\ r = -\frac{2}{3}\,\text{RootOf}(-1 + 3\,\_Z^2,\ label = \_L3)\}$$

This is not very helpful; so let's try to do more work by hand, to help Maple:

```
df1 := diff(f1,x);
xcrit := solve(df1,x);
```

$$df1 := 1 - 3\,x^2$$

$$xcrit := -\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}$$

```
> rc1 := solve(subs(x=xcrit[1],f1),r); evalf(rc1);
  rc2 := solve(subs(x=xcrit[2],f1),r); evalf(rc2);
```

$$rc1 := \frac{2\sqrt{3}}{9}$$

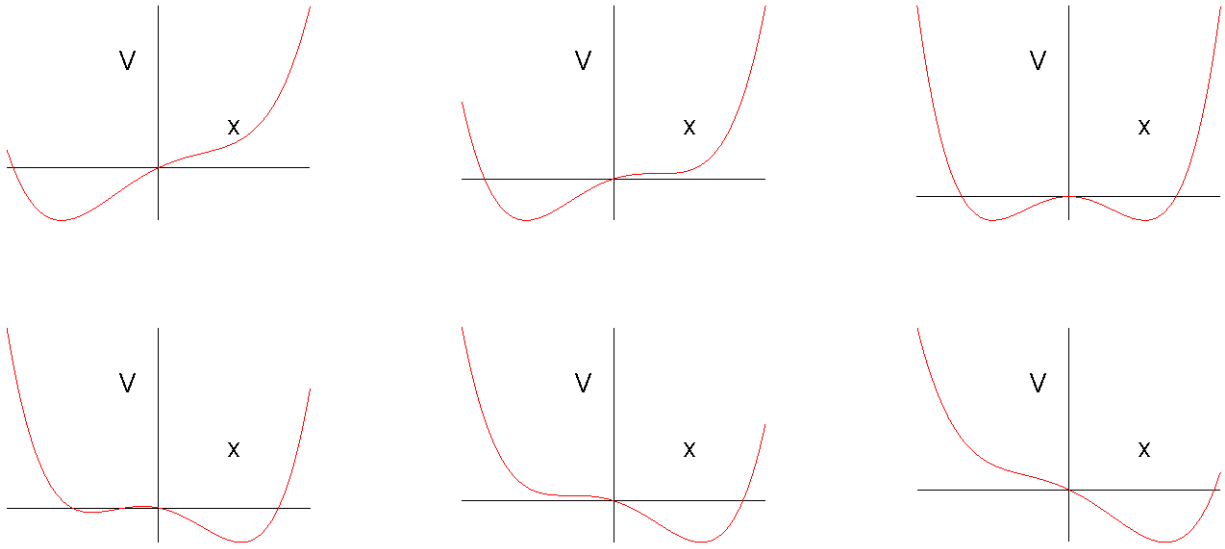
$$0.3849001795$$

$$rc2 := -\frac{2\sqrt{3}}{9}$$

$$-0.3849001795$$

Now let's plot the potential for different values of r:

```
p1a := plot(subs(r=-0.8,V1),x=-2..2,tickmarks=[0,0],labels=["x","V"]):
p1b := plot(subs(r=rc2,V1),x=-2..2,tickmarks=[0,0],labels=["x","V"]):
p1c := plot(subs(r=0,V1),x=-2..2,tickmarks=[0,0],labels=["x","V"]):
p1d := plot(subs(r=0.2,V1),x=-2..2,tickmarks=[0,0],labels=["x","V"]):
p1e := plot(subs(r=rc1,V1),x=-2..2,tickmarks=[0,0],labels=["x","V"]):
p1f := plot(subs(r=0.8,V1),x=-2..2,tickmarks=[0,0],labels=["x","V"]):
p1 := array(1..2,1..3):
p1[1,1] := p1a: p1[1,2] := p1b: p1[1,3] := p1c: p1[2,1] := p1d: p1[2,2] := p1e: p1[2,3]
:= p1f:
display(p1);
```



>

Problem 2.7.7

For the one-dimensional dynamical system $\frac{dx}{dt} = f(x)$ where $f(x) = -\frac{dV}{dx}$, the potential $V(x(t))$ evolves with the flow according to $\frac{d\,V(x(t))}{dt} = \frac{dV\,dx}{dx\,dt}$, so $\frac{dV}{dt} = -\left(\frac{dV}{dx}\right)^2$, which is <= 0. This means that the solution x(t) evolves in a way so that V(x(t)) does not increase; and it only stays constant when dV/dx=0, that is, at fixed points. For a nontrivial (non-fixed) solution, V thus has to decrease with time. If there were an oscillation, x(t) would have to get back to its original value, which requires V(x(t)) to increase again; this is not possible.

An alternative, more systematic approach: suppose the solution of $\frac{dx}{dt} = f(x)$ oscillates with some period $0 < T$, that is, $x(t+T) = x(t)$ for all t, and hence $x(T) = x(0)$, so $V(x(T)) = V(x(0))$; and suppose this is a nontrivial oscillation (not a fixed point; that is, $x(t)$ does not equal x(0) for 0<t<T). Then we have 0 = V(x(T)) - V(x(0)) = $\int_0^T \frac{dV}{dt}\,dt = -\int_0^T \left(\frac{dV}{dx}\right)^2 dt$, which is non-positive, and equals zero only if $\frac{dV}{dx} = 0$ identically, which contradicts the assumption that we are not at a fixed point.

Problem 2.8.2 (c)

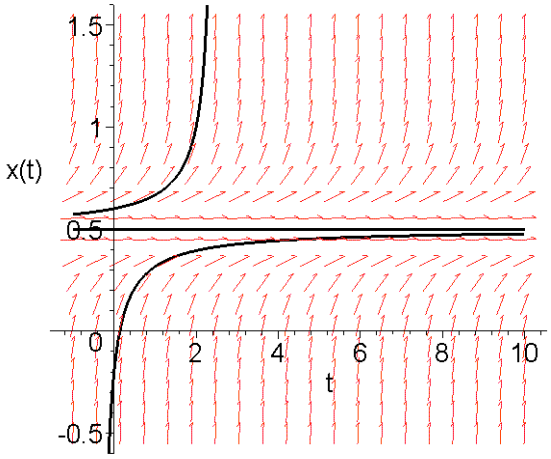
Sketching the slope field and drawing solution curves is straightforward with DEplot:

In this problem there is a half-stable fixed point at x = 1/2.

```
> deq2 := diff(x(t),t) = 1 - 4*x(t)*(1-x(t));
```

$$deq2 := \frac{d}{dt}x(t) = 1 - 4\,x(t)\,(1 - x(t))$$

```
> DEplot(deq2,x(t),t=-1..10,x=-0.5..1.5,[[x(0)=-0.2],[x(0)=0.5],[x(0)=0.6]],stepsize=0.05,linecolor=black);
```



Problems 2.8.3, 2.8.4, 2.8.5 - implementation of numerical methods

Sample code for the Euler method is available on the course website. Here is one of many possible approaches for an efficient solution, given for the improved Euler and Runge-Kutta methods (problems 2.8.4 and 2.8.5) as well as the Euler method. We first program general-purpose Euler, improved Euler and Runge-Kutta solvers, which will take any first-order ODE x'=f(t,x) as input, and give the solution at the final time tf with step h, given x(t0) = x0. Then we implement them on our particular, simple model problem.

```
> restart; with(plots):
Warning, the name changecoords has been redefined
```

```
> Euler := proc(f,t0,tf,x0,h)
    local tt, xx, i, N;
    xx := evalf(x0): tt := evalf(t0): N := evalf(ceil((tf-t0)/h));
    for i from 1 to N do
      xx := xx + h*f(tt,xx);
      tt := tt + h;
    od;
    xx;
    end:
```

```
> ImpEuler := proc(f,t0,tf,x0,h)
    local tt, xx, i, N, k;
    xx := evalf(x0): tt := evalf(t0): N := evalf(ceil((tf-t0)/h));
    for i from 1 to N do
      k  := f(tt,xx);
      xx := xx + h*(k + f(tt+h,xx+h*k))/2;
      tt := tt + h;
    od;
    xx;
    end:
```

```
> RK4 := proc(f,t0,tf,x0,h)
    local tt, xx, i, N, k1, k2, k3, k4;
    xx := evalf(x0): tt := evalf(t0): N := evalf(ceil((tf-t0)/h));
    for i from 1 to N do
      k1 := f(tt,xx);
      k2 := f(tt+h/2,xx+h*k1/2);
      k3 := f(tt+h/2,xx+h*k2/2);
      k4 := f(tt+h,xx+h*k3);
      xx := xx + h*(k1 + 2*k2 + 2*k3 + k4)/6;
      tt := tt + h;
    od;
    xx;
    end:
```

Now define the function, and find the exact solution:

```
f := (t,x) -> -x;
x0 := 'x0': soln := unapply(rhs(dsolve({diff(x(t),t) = f(t,x(t)),x(0)=x0})),t);
```

$$f := (t, x) \rightarrow -x$$

$$soln := t \rightarrow x0\, \mathbf{e}^{(-t)}$$

Exact value of x(1), given x(0) = 1:

```
evalf(subs(x0=1,soln(1)));
```

$$0.3678794412$$

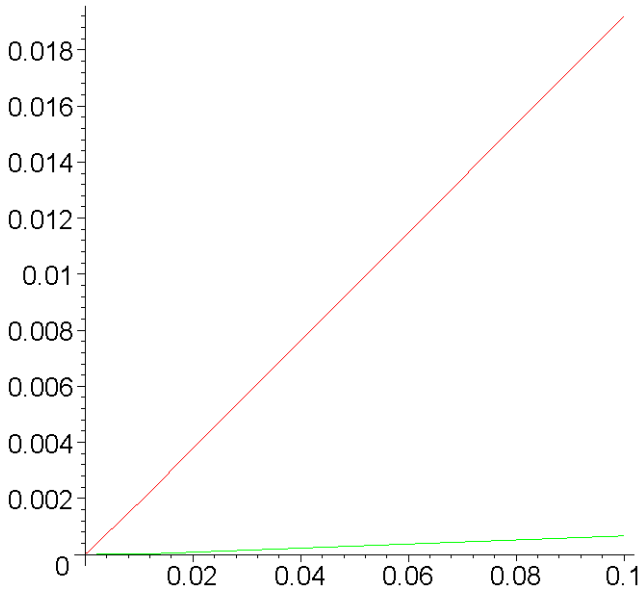
Now compute the Euler, Improved Euler and Runge-Kutta approximations to the solutions, and the errors.

```
> t0 := 0: tf := 1.: x0 := 1:
  for n from 1 to 4 do
    hval[n] := 10^(-n);
    eul[n] := Euler(f,t0,tf,x0,hval[n]);
    eul_err[n] := abs(soln(tf) - eul[n]);
    ie[n] := ImpEuler(f,t0,tf,x0,hval[n]);
    ie_err[n] := abs(soln(tf) - ie[n]);
    rk[n] := RK4(f,t0,tf,x0,hval[n]);
    rk_err[n] := abs(soln(tf) - rk[n]);
    if n = 1 then
      eul_errlist := [[hval[n],eul_err[n]]];
      ie_errlist := [[hval[n],ie_err[n]]];
      rk_errlist := [[hval[n],rk_err[n]]];
    else
      eul_errlist := [op(eul_errlist),[hval[n],eul_err[n]]];
      ie_errlist := [op(ie_errlist),[hval[n],ie_err[n]]];
      rk_errlist := [op(rk_errlist),[hval[n],rk_err[n]]];
    end if;
  od:
```

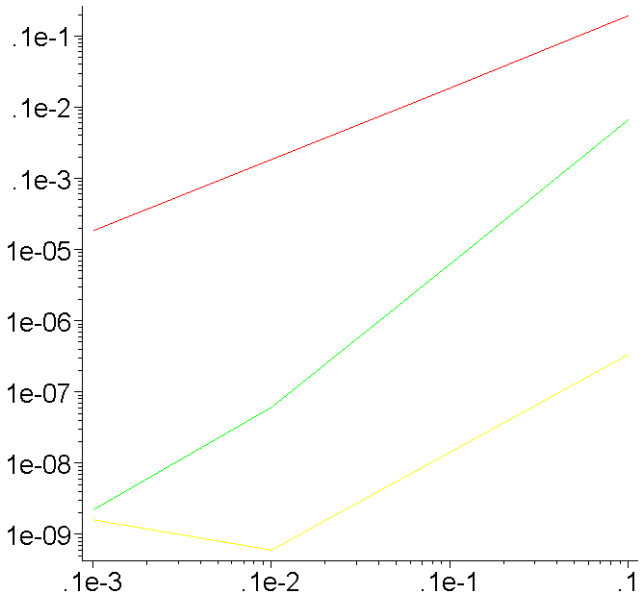
Now plot the errors:

```
plot([eul_errlist,ie_errlist,rk_errlist]);
```

Notice that on this graph, the error for improved Euler is much smaller than that for the Euler method, and the RK error is not visible. So we should plot a log-log graph:




```
> loglogplot([eul_errlist,ie_errlist,rk_errlist]);
```



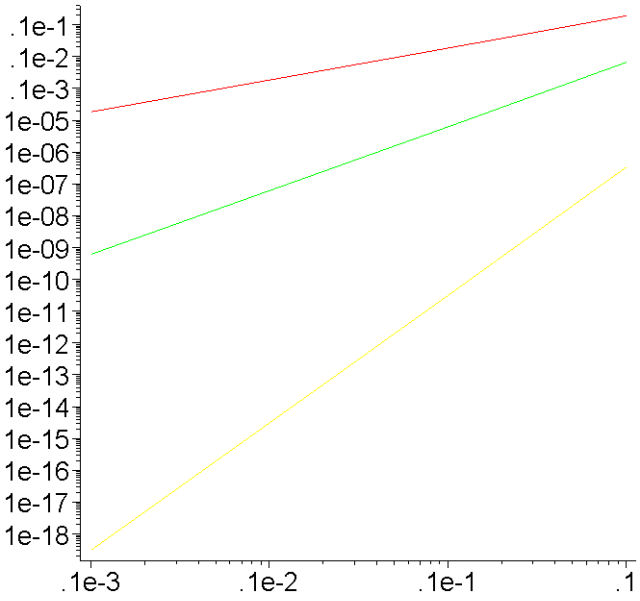
By estimating the slopes of the lines (this could be done more carefully) we see that the Euler method is first-order, and the improved Euler method is second-order. For the Runge-Kutta method, however, the error does not behave as expected; with step size 0.01, the error is computed as 0. (identically zero, to the precision used), and thus it does not show up on the log-log plot; while for smaller step size, the error increases again. This indicates that the Runge-Kutta calculation is affected by round-off error.

At the cost of increasing the computation time, we can reduce the round-off error by using more significant digits in our calculations:

```
Digits := 20;
```

Digits := 20

```
> t0 := 0: tf := 1.: x0 := 1:
  for n from 1 to 4 do
    hval[n] := 10^(-n);
    eul[n] := Euler(f,t0,tf,x0,hval[n]);
    eul_err[n] := abs(soln(tf) - eul[n]);
    ie[n] := ImpEuler(f,t0,tf,x0,hval[n]);
    ie_err[n] := abs(soln(tf) - ie[n]);
    rk[n] := RK4(f,t0,tf,x0,hval[n]);
    rk_err[n] := abs(soln(tf) - rk[n]);
    if n = 1 then
      eul_errlist := [[hval[n],eul_err[n]]];
      ie_errlist := [[hval[n],ie_err[n]]];
      rk_errlist := [[hval[n],rk_err[n]]];
    else
      eul_errlist := [op(eul_errlist),[hval[n],eul_err[n]]];
      ie_errlist := [op(ie_errlist),[hval[n],ie_err[n]]];
      rk_errlist := [op(rk_errlist),[hval[n],rk_err[n]]];
    end if;
  od:
  loglogplot([eul_errlist,ie_errlist,rk_errlist]);
```



With these higher accuracy computations, we see that the Runge-Kutta method is fourth-order, and much more accurate than the improved Euler method (the error decreased by a factor of about 10^(12) when the step size was reduced by a factor of 10^3).

Problem 3.1.3

There is a saddle-node bifurcation point at x=0, r=0, with two fixed points for r < 0 and none for r > 0.
Some representative numerical solutions (notice that the vector field is meaningless for x<=-1):

```
> restart; with(plots): with(DEtools):
Warning, the name changecoords has been redefined
```

```
> f6 := (x,r) -> r + x - ln(1+x);
```

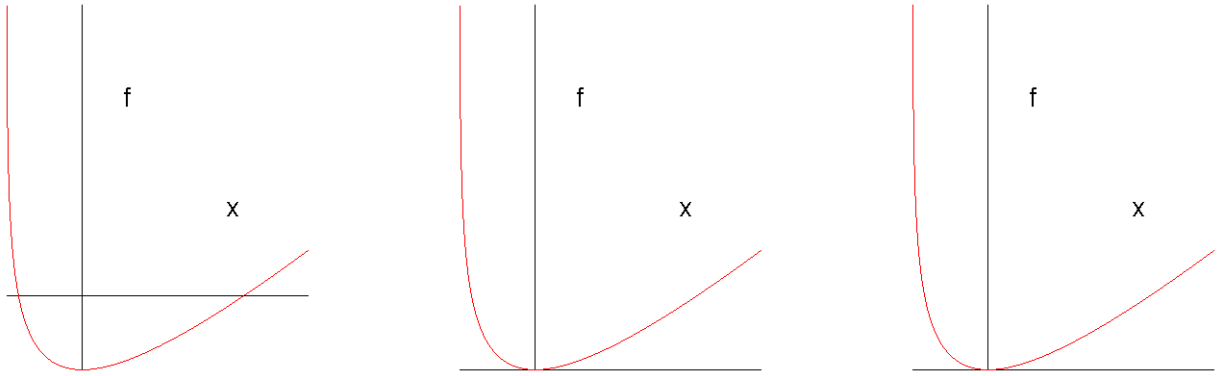
$$f6 := (x, r) \to r + x - \ln(1 + x)$$

```
> deq6 := diff(x(t),t) = f6(x(t),r);
```

$$deq6 := \frac{d}{dt}\mathrm{x}(t) = r + \mathrm{x}(t) - \ln(1 + \mathrm{x}(t))$$

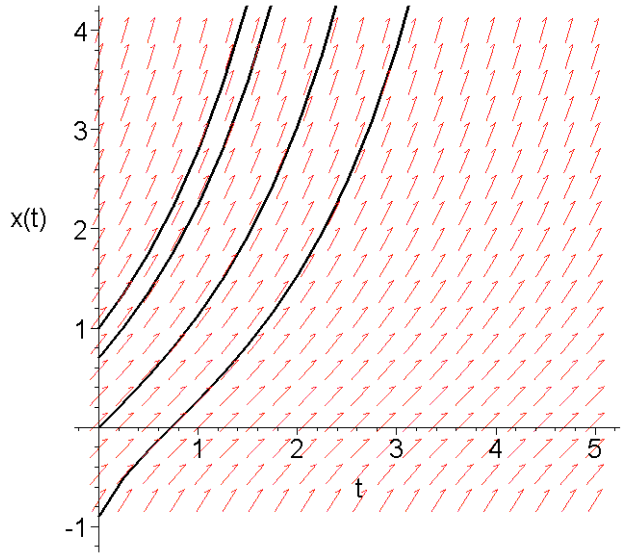
Plot qualitatively different vector fields for r<0, r=0 and r>0:

```
p6a := plot(f6(x,-1),x=-1..3,tickmarks=[0,0],labels=["x","f"]):
p6b := plot(f6(x,0),x=-1..3,tickmarks=[0,0],labels=["x","f"]):
p6c := plot(f6(x,1),x=-1..3,tickmarks=[0,0],labels=["x","f"]):
p6 := array(1..1,1..3):
p6[1,1] := p6a: p6[1,2] := p6b: p6[1,3] := p6c: display(p6);
```



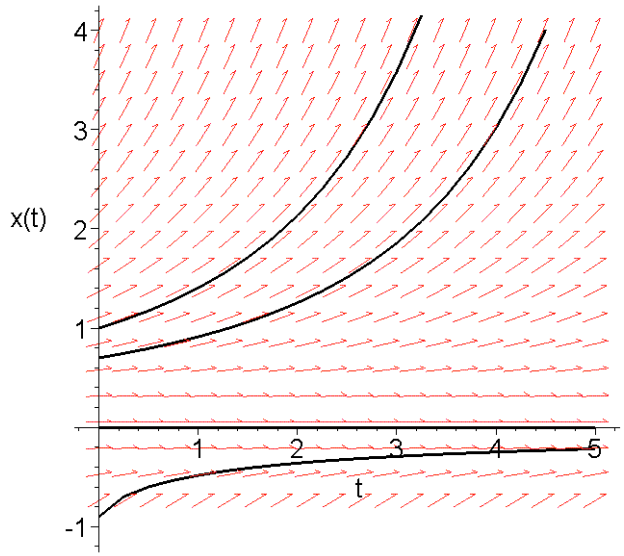

```
> DEplot(subs(r=1,deq6),x(t),t=0..5,x=-1..4,[[x(0)=-0.9],[x(0)=0],[x(0)=0.7],[x(0)=1]],l
  inecolor=black);
```

No fixed points



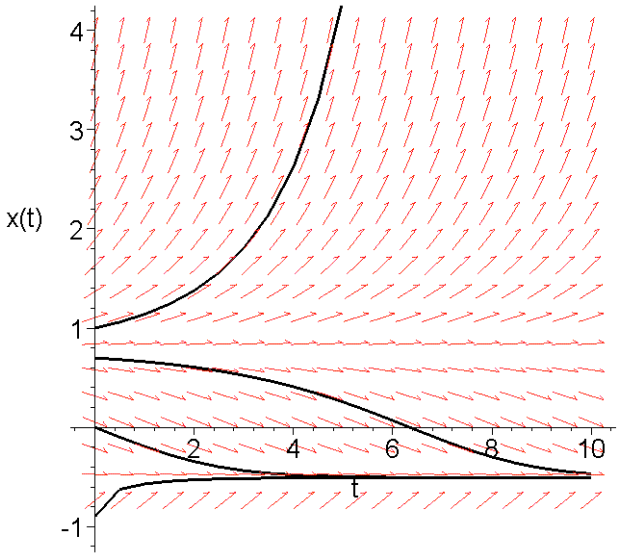

```
> DEplot(subs(r=0,deq6),x(t),t=0..5,x=-1..4,[[x(0)=-0.9],[x(0)=0],[x(0)=0.7],[x(0)=1]],l
inecolor=black);
```

One fixed point: x=0 is half-stable.



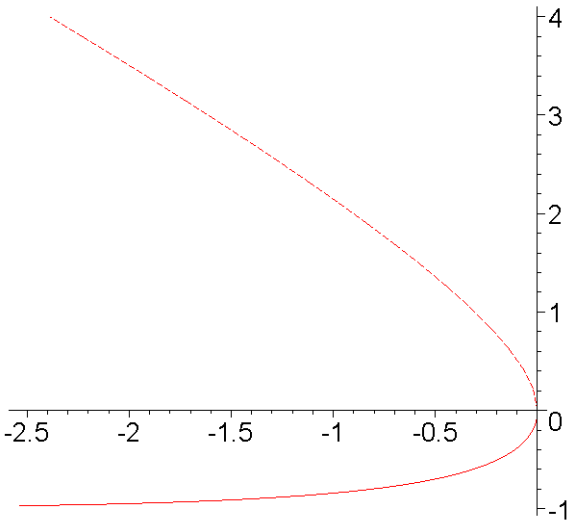

```
> DEplot(subs(r=-0.2,deq6),x(t),t=0..10,x=-1..4,[[x(0)=-0.9],[x(0)=0],[x(0)=0.7],[x(0)=1
]],linecolor=black);
```

A stable (x<0) and unstable (x>0) fixed point born in the saddle-node bifurcation at r=0.



Bifurcation diagram: (the upper branch is unstable, the lower is stable)

```
p6a := plot([-s+ln(1+s),s,s=-0.97..0],linestyle=1):
p6b := plot([-s+ln(1+s),s,s=0..4],linestyle=3,numpoints=20):
display(p6a,p6b);
```




```
>
```

Problem 3.2.4

The point x=0 is a fixed point for all r (stable for r < 1, unstable for r > 1); since there is no symmetry, we expect a transcritical bifurcation. The other fixed point is at x=ln r (which exists for r > 0, and is unstable for r < 1, stable for r > 1); there is a transcritical bifurcation at r=1.

Some representative numerical solutions:

```
> f7 := (x,r) -> x*(r - exp(x));
```

$$f7 := (x, r) \to x\,(r - \mathbf{e}^{x})$$

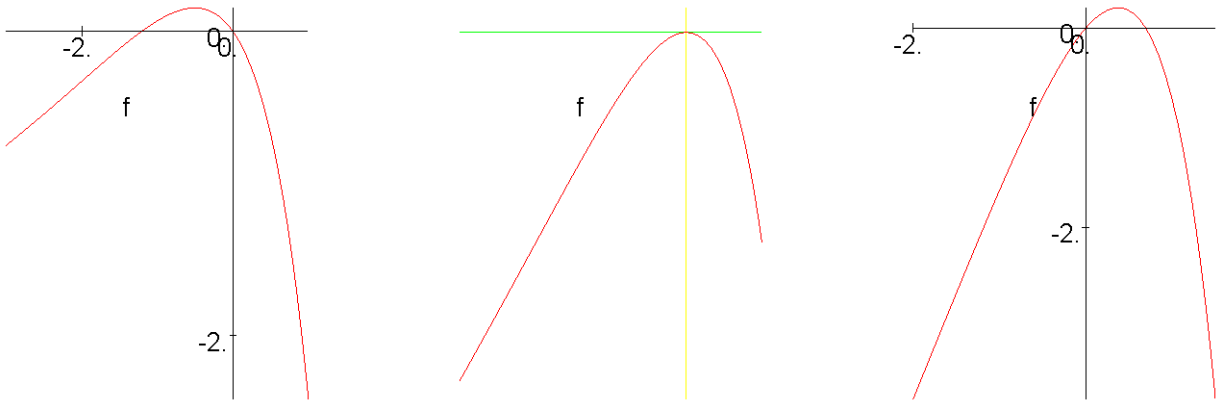
```
> deq7 := diff(x(t),t) = f7(x(t),r);
```

$$deq7 := \frac{d}{dt}\mathrm{x}(t) = \mathrm{x}(t)\,(r - \mathbf{e}^{\mathrm{x}(t)})$$

Plot qualitatively different vector fields for r<1, r=1 and r>1:

```
p7a := plot(f7(x,0.3),x=-3..1,tickmarks=[2,2],labels=["","f"]):
p7b :=
```

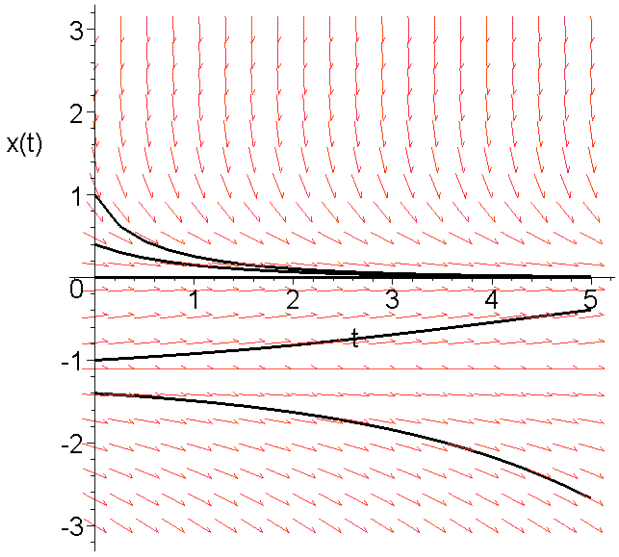
```
plot([f7(x,1),0,[[0,0.2],[0,-3]]],x=-3..1,tickmarks=[2,2],labels=["","f"],axes=none):
p7c := plot(f7(x,2),x=-2..1.5,tickmarks=[2,2],labels=["","f"]):
p7 := array(1..1,1..3):
p7[1,1] := p7a: p7[1,2] := p7b: p7[1,3] := p7c: display(p7);
>
```




```
> solve(f7(x,0.3)=0,x);
DEplot(subs(r=0.3,deq7),x(t),t=0..5,x=-3..3,[[x(0)=-1.4],[x(0)=-1],[x(0)=0],[x(0)=0.4]
,[x(0)=1]],linecolor=black);
```

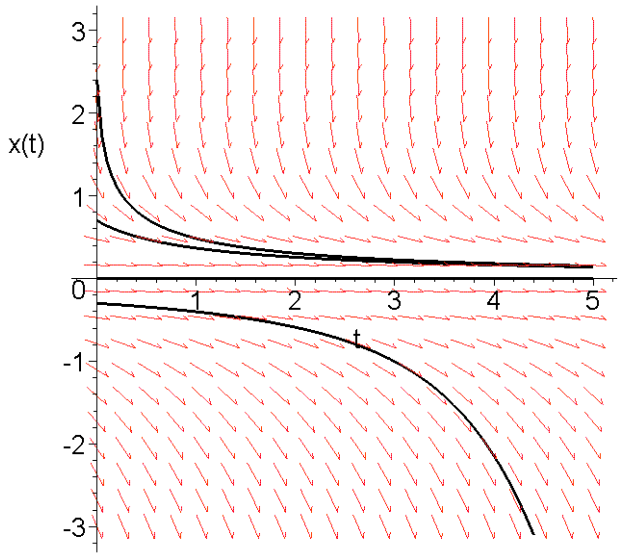
The lower fixed point is unstable, the upper is stable.

0., -1.203972804




```
> DEplot(subs(r=1,deq7),x(t),t=0..5,x=-3..3,[[x(0)=-0.3],[x(0)=0],[x(0)=0.7],[x(0)=2.4]]
,stepsize=0.05,linecolor=black);
```

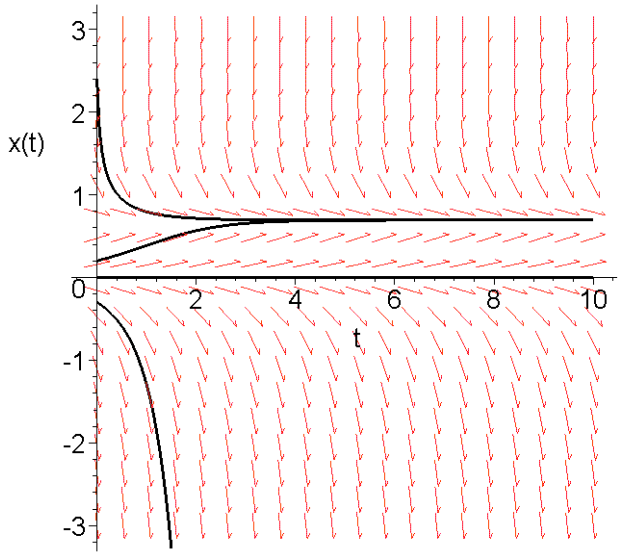
One fixed point: x=0 is half-stable.




```
> solve(f7(x,2)=0,x);
DEplot(subs(r=2,deq7),x(t),t=0..10,x=-3..3,[[x(0)=-0.3],[x(0)=0],[x(0)=0.2],[x(0)=2.4]
],stepsize=0.05,linecolor=black);
```

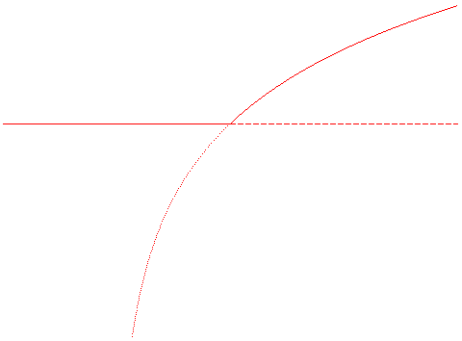
The lower fixed point x=0 is unstable, the upper is stable.

$$0,\ \ln(2)$$



Bifurcation diagram: (exchange of stabilities: the upper branch is always stable)

```
p7a := plot([[-1,0],[1,0]],linestyle=1):
p7b := plot([[1,0],[3,0]],linestyle=3):
p7c := plot(ln(r),r=0.1..1,linestyle=2,numpoints=10):
p7d := plot(ln(r),r=1..3,linestyle=1):
display(p7a,p7b,p7c,p7d,axes=none,view=[-1..3,-2..2]);
```



>

Problem 3.2.5

(a) By the Law of Mass Action, the rate at which product is formed, and reactant is used up, in a chemical reaction is proportional to the concentrations of the reactants. A discussion of how the differential equation is obtained, with an example, was found in the previous homework set, Problem 2.3.2, which you could review if you were uncertain how to derive the dynamical system. In this case, the Law of Mass Action implies (assuming that the concentrations of A and B are constant)

$\frac{dx}{dt} = k1\ a\ x - km1\ x^2 - k2\ b\ x$, which can be written in the form $\frac{dx}{dt} = c1\ x - c2\ x^2$ where $c1 = k1\ a - k2\ b$ could be positive or negative, and $c2 = km1$ is definitely positive.

(b) The fixed points are

```
solve(c1*x - c2*x^2,x);
```

$$0, \frac{c1}{c2}$$

Linear stability analysis:

```
> fp := unapply(diff(c1*x - c2*x^2,x),x);
> fp(0); fp(c1/c2);
```

$$fp := x \rightarrow c1 - 2\ c2\ x$$

$$c1$$

$$-c1$$

Thus the fixed point at x=0 is stable for c1<0 and unstable for c1>0, while the other fixed point is stable for c1>0. That is, the fixed point x=0 is stable if k1*a < k2*b. This makes sense chemically: k1*a is the rate at which X is produced by the first reaction, while k2*b is the rate at which it is consumed by the second reaction, so if k2*b > k1*a, the rate of consumption is greater than the rate of production, the chemical X is used up as rapidly as it can be created, and thus X will remain at the equilibrium of zero concentration.

Problem 3.2.6

We can use computer algebra to aid us in the normal form calculation: Following the steps suggested in the problem -

```
> x1 := series(X + b*X^3 + b2*X^4,X=0,4);
```

$$x1 := X + b\ X^3 + O(X^4)$$

Solve for X in terms of x:

```
> sol1 := solve(series(x1,X) = x, X);
```

Note that if we hadn't included a fourth-order term in the above definition of the series x1, at this point Maple would have seen a cubic equation which it can solve exactly, so it would have presented the exact, but very complicated-looking, formula for roots of a cubic, instead of the approximation for X near 0 which we seek.

$$sol1 := x - b\ x^3 + O(x^4)$$

Definition of the original dynamical system:

```
> Xdot := series(R*X - X^2 + a*X^3 + a2*X^4,X=0,4);
```

$$Xdot := R\ X - X^2 + a\ X^3 + O(X^4)$$

We need to convert the Maple 'series' form to polynomials to be able to perform the substitutions and multiplications (better: use the formal power series package, see below):

```
> p1 := convert(x1,polynom);
  pXdot := convert(Xdot,polynom);
```

```
> psol1 := convert(sol1,polynom);
```

$$p1 := X + b\,X^3$$

$$pXdot := R\,X - X^2 + a\,X^3$$

$$psol1 := x - b\,x^3$$

We wish to find the dynamical system in terms of the new variable x. So apply the chain rule, dx/dt = dx/dX dX/dt

```
> p1dot := unapply(collect(diff(p1,X)*pXdot,X),X);
```

$$p1dot := X \to 3\,b\,a\,X^5 - 3\,b\,X^4 + (a + 3\,b\,R)\,X^3 - X^2 + R\,X$$

and substitute X in terms of x:

```
> pxdot := p1dot(psol1);
```

$$pxdot := 3\,b\,a\,(x - b\,x^3)^5 - 3\,b\,(x - b\,x^3)^4 + (a + 3\,b\,R)\,(x - b\,x^3)^3 - (x - b\,x^3)^2 + R\,(x - b\,x^3)$$

```
> collect(expand(pxdot),x);
```

$$-3\,b^6\,a\,x^{15} + 15\,b^5\,a\,x^{13} - 3\,b^5\,x^{12} - 30\,b^4\,a\,x^{11} + 12\,b^4\,x^{10} + (29\,b^3\,a - 3\,b^4\,R)\,x^9 - 18\,b^3\,x^8 + (-12\,b^2\,a + 9\,b^3\,R)\,x^7 + 11\,b^2\,x^6 - 9\,b^2\,R\,x^5 - b\,x^4 + (a + 2\,b\,R)\,x^3 - x^2 + R\,x$$

```
> series(%,x=0,4);
```

$$R\,x - x^2 + (a + 2\,b\,R)\,x^3 + O(x^4)$$

Now pxdot contains the new dynamical system in terms of x. The linear and quadratic terms are the same as for the original dynamical system in terms of X, and at this point, we can choose b in the near-identity transformation so that the coefficient of the cubic term vanishes: Use coeftayl to extract the coefficient, and solve for b.

```
> eq3 := coeftayl(pxdot,x=0,3);
```

$$eq3 := a + 2\,b\,R$$

```
> bsol := solve(eq3,b);
```

$$bsol := -\frac{a}{2\,R}$$

```
> series(expand(subs(b=bsol,pxdot)),x=0,4);
```

$$R\,x - x^2 + O(x^4)$$

It is necessary to make the assumption that R is nonzero; if R=0, we cannot remove any higher-order terms.

```
>
```

Formal power series method

An alternative approach: instead of converting the series to polynomials, we can do all the calculations using the Maple package for the manipulation of formal power series. The tpsform command evaluates the series to the chosen order. The steps are the same as above; this time we do all calculations to fourth order, eliminating the first two higher-order terms; by problem 3.2.7, we can eliminate any higher-order term.

```
> restart;  with(powseries);
```

[*compose, evalpow, inverse, multconst, multiply, negative, powadd, powcos, powcreate, powdiff, powexp, powint, powlog, powpoly, powsin, powsolve, powsqrt, quotient, reversion, subtract, template, tpsform*]

```
>
> p1 := powpoly(X + b*X^3 + b2*X^4 + b3*X^5,X);
```

p1 := **proc**(*powparm*) ... **end proc**

```
> s1 := tpsform(p1,X,5);
```

The near-identity transformation: x in terms of X

$$s1 := X + b\,X^3 + b2\,X^4 + O(X^5)$$

```
> p2 := reversion(p1);
```

p2 := **proc**(*powparm*) ... **end proc**

```
> s2 := tpsform(p2,x,5);
```

X in terms of x, reverting the original power series p1

$$s2 := x - b\,x^3 - b2\,x^4 + O(x^5)$$

```
> p3 := powdiff(p1);
```

p3 := **proc**(*powparm*) ... **end proc**

```
> pXdot := powpoly(R*X - X^2 + a*X^3 + a2*X^4,X);
```

pXdot := **proc**(*powparm*) ... **end proc**

```
> p4 := multiply(p3,pXdot);
```

p4 := **proc**(*powparm*) ... **end proc**

```
> s4 := tpsform(p4,X,5);
```

dx/dt in terms of X, computed by the chain rule dx/dt = dx/dX dX/dt

$$s4 := R\,X - X^2 + (a + 3\,b\,R)\,X^3 + (a2 - 3\,b + 4\,b2\,R)\,X^4 + O(X^5)$$

```
> p5 := compose(p4,p2);
```

$$p5 := \mathbf{proc}(powparm) \ \ldots \ \mathbf{end\ proc}$$

```
> s5 := tpsform(p5,x,5);
```

Now write dx/dt in terms of x, composing p4(X) with the power series p2 giving X(x)

$$s5 := R\,x - x^2 + (2\,b\,R + a)\,x^3 + (3\,b2\,R - b + a2)\,x^4 + O(x^5)$$

Now extract the coefficients of the cubic and quartic terms, and solve for b and b2:

```
> coef3 := coeftayl(s5,x=0,3);
  coef4 := coeftayl(s5,x=0,4);
```

$$coef3 := 2\,b\,R + a$$

$$coef4 := 3\,b2\,R - b + a2$$

```
> bsol := solve(coef3,b);
```

$$bsol := -\frac{a}{2\,R}$$

```
> b2sol := solve(subs(b=bsol,coef4),b2);
```

$$b2sol := -\frac{a + 2\,a2\,R}{6\,R^2}$$

```
> s6 := simplify(subs({b=bsol,b2=b2sol},s5));
```

$$s6 := R\,x - x^2 + O(x^5)$$

Continuing in this way, we can eliminate as many higher-order terms as we like, provided R is nonzero.