SHRI RAMDEOBABA COLLEGE OF ENGINNERING AND MANAGEMENT

NAME:TINA BORUNDIA

BATCH:C3

ROLL N0:65

EXPERIMENT NO:6

AIM:Solution of differential equations in SAGEMATH

In [1]:
```
x=var('x')
```

In [2]:
```
y=function('y')(x)
```

In [3]:
```
show(desolve(diff(y,x)-sin(2*x)==0,y,show_method=True))
```

$$\left[C - \frac{1}{2}\cos(2\,x), \mathtt{linear}\right]$$

In [4]:
```
show(desolve(diff(y,x)==sin(2*x),y))
```

$$C - \frac{1}{2}\cos(2\,x)$$

In [5]:
```
show(desolve(diff(y,x)-sin(2*x),y,x))
```

$$C - \frac{1}{2}\cos(2\,x)$$

In [6]:
```
desolve(diff(y, x)-sin(2*x),y,[0,1], show_method=True)
```

Out[6]:
```
[-1/2*cos(2*x) + 3/2, 'linear']
```

In [ ]:
```
y=-1/2*cos(2*x) + 3/2
plot(y,(x,-2*pi,2*pi),figsize=3)
```

In [1]:
```
x=var('x')
y=function('y')(x)
desolve(diff(y,x)-x,y,show_method=True)
```

Out[1]:
```
[1/2*x^2 + _C, 'linear']
```

In [ ]:
```
P=Graphics()
for C in srange(-5,5,1.5):
    y=(x^2)/(2+C)
P=P+plot(y,(x,-2*pi,2*pi), color='red',figsize=3)
P
```

In [7]:
```
x=var('x')
```

In [8]:
```
y=function('y')(x)
```

In [9]:
```
desolve(diff(y,x)-(x*y^2)+1/x*y,y)
```

Out[9]:
```
1/((_C - x)*x)
```

In [10]:
```
desolve(diff(y,x)+1/x*y==x*y^2, y, show_method=True)
```

Out[10]:
```
[1/((_C - x)*x), 'bernoulli']
```

In [4]:
```
x=var('x')
y=function('y')(x)
f = desolve(diff(y,x) + y-1, y, [1,2])
show(f)
```

$$(e + e^{x})e^{(-x)}$$

In [ ]:
```
plot(f,(x,-2,2),color='red',figsize=3)
```

In [ ]:
```
f = desolve(diff(y,x) + y-1, y, [1,2])
f
```

In [ ]:
```
show(f)
```

In [ ]:
```
plot(f,(x,-2,2),color='red', figsize=3)
```

In [ ]:
```
y=var('y')
p1=plot_slope_field(1-y,(x,0,3),(y,0,5))
p2=plot(f, (x, 0,3), color='red', figsize=4)
p1+p2
```

In [2]:
```
help(srange)
```

```
Help on built-in function srange in module sage.arith.srange:

srange(...)
    srange(*args, **kwds)
    File: sage/arith/srange.pyx (starting at line 177)

        Return a list of numbers
        ``start, start+step, ..., start+k*step``,
        where ``start+k*step < end`` and ``start+(k+1)*step >= end``.

        This provides one way to iterate over Sage integers as opposed to
        Python int's.  It also allows you to specify step sizes for such
        an iteration.

        INPUT:

        - ``start`` - number (default: 0)

        - ``end`` - number

        - ``step`` - number (default: 1)

        - ``universe -- parent or type where all the elements should live
          (default: deduce from inputs). This is only used if ``coerce`` is
          true.

        - ``coerce`` -- convert ``start``, ``end`` and ``step`` to the same
          universe (either the universe given in ``universe`` or the
          automatically detected universe)

        - ``include_endpoint`` -- whether or not to include the endpoint
          (default: False). This is only relevant if ``end`` is actually of
          the form ``start + k*step`` for some integer `k`.

        ` ``endpoint_tolerance`` -- used to determine whether or not the
          endpoint is hit for inexact rings (default 1e-5)

        OUTPUT: a list

        .. NOTE::

           This function is called ``srange`` to distinguish
           it from the built-in Python ``range`` command.  The s
           at the beginning of the name stands for "Sage".

        .. SEEALSO::

            :func:`xsrange` -- iterator which is used to implement :func:`sra
nge`.

        EXAMPLES::

            sage: v = srange(5); v
            [0, 1, 2, 3, 4]
            sage: type(v[2])
            <type 'sage.rings.integer.Integer'>
            sage: srange(1, 10)
```

```
            [1, 2, 3, 4, 5, 6, 7, 8, 9]
            sage: srange(10, 1, -1)
            [10, 9, 8, 7, 6, 5, 4, 3, 2]
            sage: srange(10,1,-1, include_endpoint=True)
            [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
            sage: srange(1, 10, universe=RDF)
            [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0]

            sage: srange(1, 10, 1/2)
            [1, 3/2, 2, 5/2, 3, 7/2, 4, 9/2, 5, 11/2, 6, 13/2, 7, 15/2, 8, 1
7/2, 9, 19/2]
            sage: srange(1, 5, 0.5)
            [1.00000000000000, 1.50000000000000, 2.00000000000000, 2.50000000
000000, 3.00000000000000, 3.50000000000000, 4.00000000000000, 4.5000000000000
0]
            sage: srange(0, 1, 0.4)
            [0.000000000000000, 0.400000000000000, 0.800000000000000]
            sage: srange(1.0, 5.0, include_endpoint=True)
            [1.00000000000000, 2.00000000000000, 3.00000000000000, 4.00000000
000000, 5.00000000000000]
            sage: srange(1.0, 1.1)
            [1.00000000000000]
            sage: srange(1.0, 1.0)
            []
            sage: V = VectorSpace(QQ, 2)
            sage: srange(V([0,0]), V([5,5]), step=V([2,2]))
            [(0, 0), (2, 2), (4, 4)]

        Including the endpoint::

            sage: srange(0, 10, step=2, include_endpoint=True)
            [0, 2, 4, 6, 8, 10]
            sage: srange(0, 10, step=3, include_endpoint=True)
            [0, 3, 6, 9]

        Try some inexact rings::

            sage: srange(0.5, 1.1, 0.1, universe=RDF, include_endpoint=False)
            [0.5, 0.6, 0.7, 0.7999999999999999, 0.8999999999999999, 0.9999999
999999999]
            sage: srange(0.5, 1, 0.1, universe=RDF, include_endpoint=False)
            [0.5, 0.6, 0.7, 0.7999999999999999, 0.8999999999999999]
            sage: srange(0.5, 0.9, 0.1, universe=RDF, include_endpoint=False)
            [0.5, 0.6, 0.7, 0.7999999999999999]
            sage: srange(0, 1.1, 0.1, universe=RDF, include_endpoint=True)
            [0.0, 0.1, 0.2, 0.30000000000000004, 0.4, 0.5, 0.6, 0.7, 0.799999
9999999999, 0.8999999999999999, 0.9999999999999999, 1.1]
            sage: srange(0, 0.2, 0.1, universe=RDF, include_endpoint=True)
            [0.0, 0.1, 0.2]
            sage: srange(0, 0.3, 0.1, universe=RDF, include_endpoint=True)
            [0.0, 0.1, 0.2, 0.3]

        More examples::

            sage: Q = RationalField()
            sage: srange(1, 10, Q('1/2'))
            [1, 3/2, 2, 5/2, 3, 7/2, 4, 9/2, 5, 11/2, 6, 13/2, 7, 15/2, 8, 1
```

```
7/2, 9, 19/2]
            sage: srange(1, 5, 0.5)
            [1.00000000000000, 1.50000000000000, 2.00000000000000, 2.50000000
000000, 3.00000000000000, 3.50000000000000, 4.00000000000000, 4.5000000000000
0]
            sage: srange(0, 1, 0.4)
            [0.000000000000000, 0.400000000000000, 0.800000000000000]

        Negative steps are also allowed::

            sage: srange(4, 1, -1)
            [4, 3, 2]
            sage: srange(4, 1, -1/2)
            [4, 7/2, 3, 5/2, 2, 3/2]

        TESTS:

        These are doctests from :trac:`6409`::

            sage: srange(1,QQ(0),include_endpoint=True)
            []
            sage: srange(1,QQ(0),-1,include_endpoint=True)
            [1, 0]

        Test :trac:`11753`::

            sage: srange(1,1,0)
            Traceback (most recent call last):
            ...
            ValueError: step argument must not be zero

        No problems with large lists::

            sage: srange(10^5) == list(range(10^5))
            True
```

In [4]:
```
y=function('y')(x)
desolve(diff(y,x)+(y)==cos(x),y)
```

Out[4]: `1/2*((cos(x) + sin(x))*e^x + 2*_C)*e^(-x)`

In [5]:
```
desolve(diff(y,x)+(y)==cos(x),y)
```

Out[5]: `1/2*((cos(x) + sin(x))*e^x + 2*_C)*e^(-x)`

In [6]:
```
show(desolve(diff(y,x)+(y)
==
cos(x),y,show_method=True))
```

$$\left[\frac{1}{2}\left((\cos(x)+\sin(x))e^{x}+2\,C\right)e^{(-x)}, \texttt{linear}\right]$$

In [7]:
```
show(desolve(diff(y,x)+(y)
==
cos(x),y,[0,1]))
```

$$\frac{1}{2}\left(\cos(x)e^{x} + e^{x}\sin(x) + 1\right)e^{(-x)}$$

In [ ]:
```
f=desolve(diff(y,x)+(y) == cos(x),y,[0,1])
plot(f)
```

ELECTRI CIRCUIT

In [5]:
```
var('t')
i=function('i')(t)
R=2
L=40
E=20
de=desolve(diff(i,t)+(R/L)*i-(E/L),i,[0,0])
show(de)
```

$$10\left(e^{\left(\frac{1}{20}t\right)} - 1\right)e^{\left(-\frac{1}{20}t\right)}$$

In [ ]:
```
plot(de,0,1000, figsize=4)
```

EXERCISE 4.1

In [2]:
```
x=var('x')
y=function('y')(x)
show(desolve(diff(y,x,2)+2*diff(y,x)+y==cos(x),y,show_method=True))
```

$$\left[(K_2 x + K_1)e^{(-x)} + \frac{1}{2}\sin(x), \texttt{variationofparameters}\right]$$

In [ ]:
```
f=desolve(diff(y,x,2)+2*diff(y,x)+y==cos(x),y,[0,3,pi/2,2])
show(f)
plot(f,(x,-1,50), figsize=3)
```

$$3\left(\frac{x\left(e^{\left(\frac{1}{2}\pi\right)} - 2\right)}{\pi} + 1\right)e^{(-x)} + \frac{1}{2}\sin(x)$$

In [ ]:
```
show(desolve(diff(y,x,2)+2*diff(y,x)+y == cos(x)
             ,y,[0,3,pi/2,2]))
```

In [ ]:
```
f=desolve(diff(y,x,2)+2*diff(y,x)+y==cos(x),
          y,[0,3,pi/2,2])
plot(f,(x,-1,4),figsize=3)
```

EXERCISE 4.2

In [ ]:
```
t=var('t')
X = function('x')(t)
y = function('y')(t)
de1 = diff(x,t) + y - 1 == 0
de2 = diff(y,t) - x + 1 == 0
desolve_system([de1, de2], [x,y], ics=[0,1,2])
```

In [ ]:
```
x(t)= -sin(t) +1
y(t)=cos(t) +1
p1=plot(x(t), -2*pi, 2*pi, color='red',
        title="Simultaneous equation",
        legend_label="solution of x(t)")
p2=plot(y(t), -2*pi, 2*pi,
        legend_label="solution of y(t)")
p1+p2
```

EXERCISE 4.3

In [2]:
```
t = var('t')
x = function('x')(t)
y = function('y')(t)
eq1 = diff(x, t) + 2*x - 3*y == 0
eq2 = diff(y, t) - 3*x + 2*y == 0
sol = desolve_system([eq1, eq2], [x, y], ivar=t)
sol
```

Out[2]:
```
[x(t) == 1/2*(x(0) - y(0))*e^(-5*t) + 1/2*(x(0) + y(0))*e^t,
 y(t) == -1/2*(x(0) - y(0))*e^(-5*t) + 1/2*(x(0) + y(0))*e^t]
```

In [ ]:
```
t = var('t')
i1 = function('i1')(t)
i2 = function('i2')(t)
eq1 = diff(i1, t) + i2 == sin(t)
eq2 = diff(i2, t) + i2 == cos(t)
ics = [i1(0) == 2, i2(0) == 0]
sol = desolve_system([eq1, eq2], [i1, i2], ivar=t, ics=ics)
i1_sol = sol[0][0].rhs()
i2_sol = sol[1][0].rhs()
P = Graphics()
P += plot(i1_sol, (t, 0, 10), color='blue', legend_label='i1')
P += plot(i2_sol, (t, 0, 10), color='red', legend_label='i2')
P.show()
```

I have learned to implement solution of differential equations in SAGEMATH.

In [ ]: