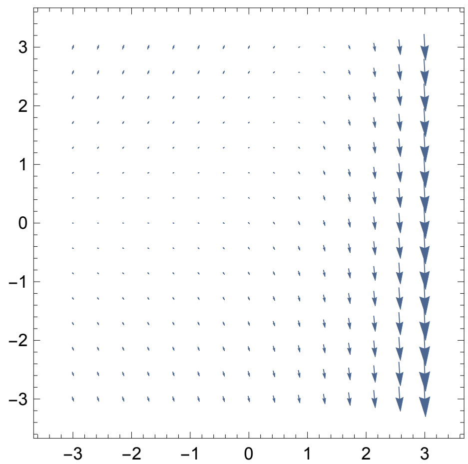
Slope Fields

We will consider the differential equation dy/dt = y - e^t

The standard VectorPlot command doesn't produce a very nice slope field:

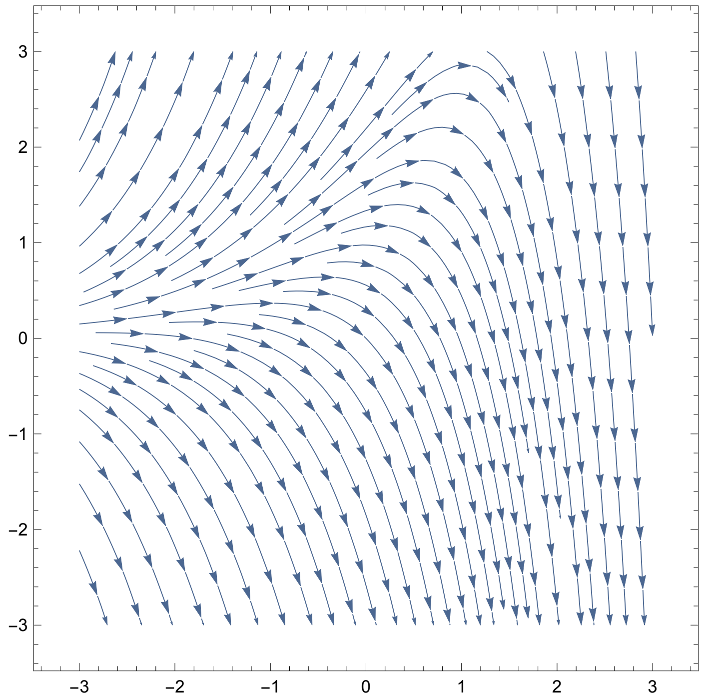
```
In[4]:= VectorPlot[{1, y - E^t}, {t, -3, 3}, {y, -3, 3}]
```



The StreamPlot command gives a better picture of the family of solutions to the differential equation:

In[7]:=
```
StreamPlot[{1, y - E^t}, {t, -3, 3}, {y, -3, 3}]
```

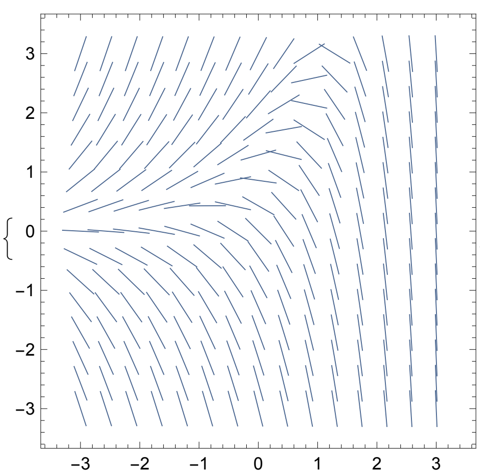
Out[7]=



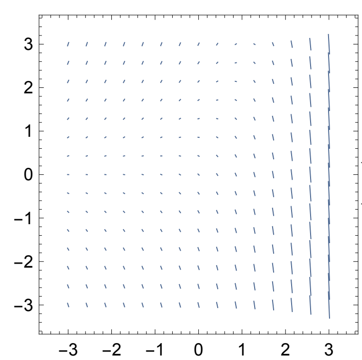
Thanks to Jacob Vincent for finding a way to draw a slope field in *Mathematica:*

In[1]:=
```
Table[VectorPlot[{1, y - E^t}, {t, -3, 3},
  {y, -3, 3}, VectorScale → {Automatic, Automatic, f},
  VectorStyle → "Segment"], {f, {None, Automatic}}]
```

Out[1]= {



,



}

Euler's Method

Given dy/dt = 2y - sin(t), with y(0)=3, we will use Euler's method to approximate y(1).

Using a Do loop:

```
In[8]:= f[t_, y_] = 2 * y - Sin[t]
        y[0] = 3
        Δt = 0.01

Out[8]= 2 y - Sin[t]

Out[9]= 3

Out[10]= 0.01
```

```
In[11]:= Do[y[n + 1] = f[Δt * n, y[n]] * Δt + y[n], {n, 0, 99}]
```

Note that after running the above line, y[100] contains the value of y after 100 steps, which is our approximation of y(1).

```
In[12]:= y[100]

Out[12]= 20.7345
```

Using the NDSolve command:

```
In[20]:= Clear[y]
         y = y /. First[NDSolve[{y'[t] == f[t, y[t]], y[0] == 3},
              y, {t, 0, 1}, StartingStepSize → 0.01, Method → "ExplicitEuler"]]

Out[21]= InterpolatingFunction[ Domain: {{0., 1.}}
                                Output: scalar ]
```

```
In[25]:= y[1]

Out[25]= 20.7345
```

Finding the actual solution using DSolve:

```
In[27]:= Clear[y]
         a = DSolve[{y'[t] == f[t, y[t]], y[0] == 3}, y[t], t]
```

Out[28]= $\left\{\left\{y[t] \to \frac{1}{5}\left(14\, e^{2t} + \mathrm{Cos}[t] + 2\,\mathrm{Sin}[t]\right)\right\}\right\}$

In[29]:=
```
Evaluate[y[t] /. a /. t → 1]
```

Out[29]= $\left\{\frac{1}{5}\left(14\, e^{2}+\text{Cos}[1]+2\,\text{Sin}[1]\right)\right\}$

In[30]:=
```
N[%]
```

Out[30]= {21.134}

Plotting the approximate and exact solutions:

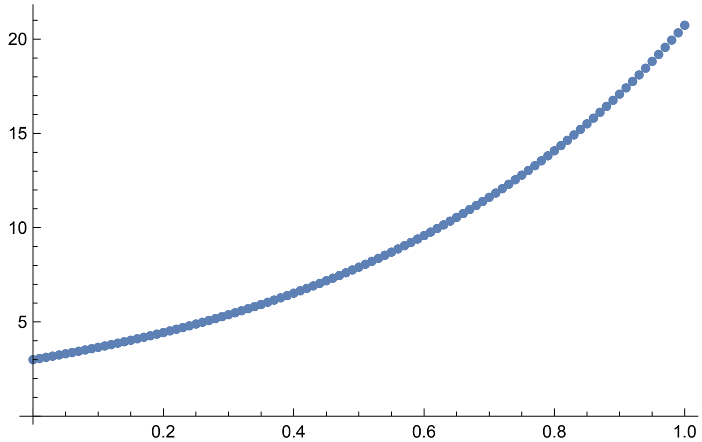
In[31]:=
```
Clear[y]
y[0] = 3
Do[y[n + 1] = f[Δt * n, y[n]] * Δt + y[n], {n, 0, 99}]
points = Table[{Δt * n, y[n]}, {n, 0, 100}]
```

Out[32]= 3

Out[34]=
```
{{0., 3}, {0.01, 3.06}, {0.02, 3.1211}, {0.03, 3.18332}, {0.04, 3.24669},
 {0.05, 3.31122}, {0.06, 3.37695}, {0.07, 3.44389}, {0.08, 3.51206}, {0.09, 3.58151},
 {0.1, 3.65224}, {0.11, 3.72428}, {0.12, 3.79767}, {0.13, 3.87243}, {0.14, 3.94858},
 {0.15, 4.02616}, {0.16, 4.10519}, {0.17, 4.1857}, {0.18, 4.26772}, {0.19, 4.35128},
 {0.2, 4.43642}, {0.21, 4.52316}, {0.22, 4.61154}, {0.23, 4.70159}, {0.24, 4.79334},
 {0.25, 4.88683}, {0.26, 4.98209}, {0.27, 5.07916}, {0.28, 5.17808},
 {0.29, 5.27888}, {0.3, 5.3816}, {0.31, 5.48627}, {0.32, 5.59295}, {0.33, 5.70166},
 {0.34, 5.81245}, {0.35, 5.92537}, {0.36, 6.04045}, {0.37, 6.15773},
 {0.38, 6.27727}, {0.39, 6.39911}, {0.4, 6.52329}, {0.41, 6.64986}, {0.42, 6.77887},
 {0.43, 6.91037}, {0.44, 7.04441}, {0.45, 7.18104}, {0.46, 7.32031},
 {0.47, 7.46228}, {0.48, 7.60699}, {0.49, 7.75451}, {0.5, 7.9049}, {0.51, 8.0582},
 {0.52, 8.21448}, {0.53, 8.3738}, {0.54, 8.53623}, {0.55, 8.70181}, {0.56, 8.87062},
 {0.57, 9.04272}, {0.58, 9.21818}, {0.59, 9.39706}, {0.6, 9.57944}, {0.61, 9.76538},
 {0.62, 9.95496}, {0.63, 10.1482}, {0.64, 10.3453}, {0.65, 10.5463}, {0.66, 10.7511},
 {0.67, 10.96}, {0.68, 11.173}, {0.69, 11.3902}, {0.7, 11.6116}, {0.71, 11.8374},
 {0.72, 12.0676}, {0.73, 12.3024}, {0.74, 12.5418}, {0.75, 12.7859}, {0.76, 13.0348},
 {0.77, 13.2886}, {0.78, 13.5474}, {0.79, 13.8113}, {0.8, 14.0804}, {0.81, 14.3549},
 {0.82, 14.6347}, {0.83, 14.9201}, {0.84, 15.2111}, {0.85, 15.5079},
 {0.86, 15.8105}, {0.87, 16.1192}, {0.88, 16.4339}, {0.89, 16.7549}, {0.9, 17.0822},
 {0.91, 17.416}, {0.92, 17.7565}, {0.93, 18.1036}, {0.94, 18.4577}, {0.95, 18.8188},
 {0.96, 19.187}, {0.97, 19.5625}, {0.98, 19.9455}, {0.99, 20.3362}, {1., 20.7345}}
```

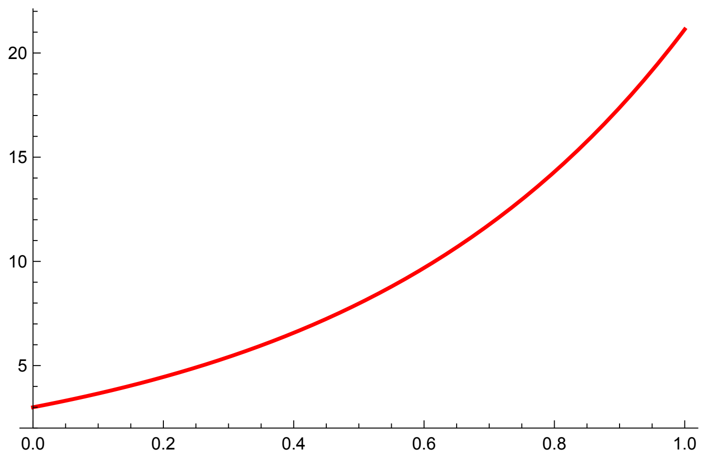
In[35]:=
```
EulerPlot = ListPlot[points]
```

Out[35]=



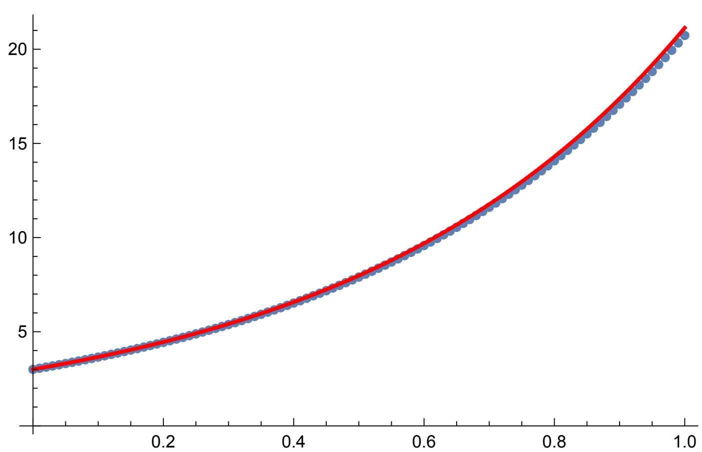
In[36]:=
```
ActualPlot =
  Plot[(14 E^(2 t) + Cos[t] + 2 Sin[t]) / 5, {t, 0, 1}, PlotStyle → {Thick, Red}]
```

Out[36]=



In[37]:=
```
Show[EulerPlot, ActualPlot]
```

Out[37]=



Using Wolfram Alpha:

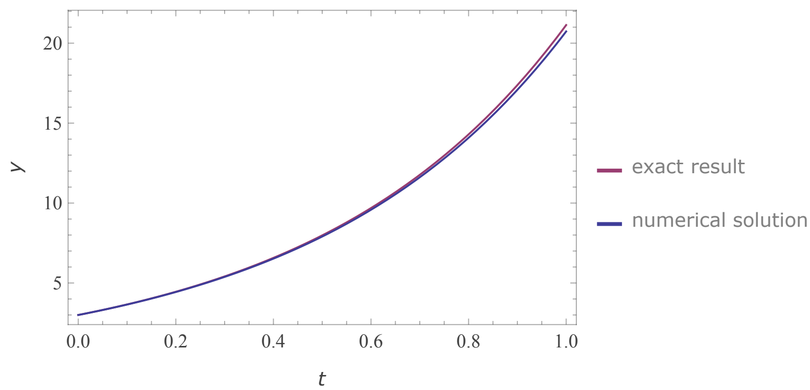
In[38]:= **use Euler method y'=2y-Sin[t], y(0)=3, from 0 to 1, stepsize 0.01**

Input interpretation:

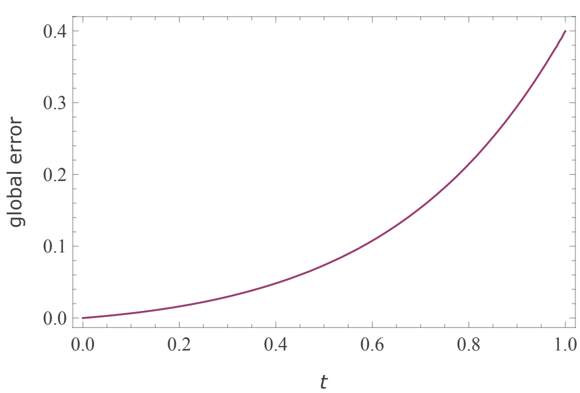
solve	$y'(t) = -\sin(t) + 2\,y(t)$ $y(0) = 3$	using Euler method	with a stepsize of 0.01 from $y = 0$ to 1

Solution plot:

Hide error plot



Error plot:



Stepwise results:

Less More

step	t	y	local error	global error
0	0.	3.	0.	0.
5	0.05	3.31122	0.000599848	0.00299791
10	0.1	3.65224	0.000663113	0.0066238
15	0.15	4.02616	0.000733163	0.0109769

20	0.2	4.43642	0.00081071	0.0161704
25	0.25	4.88683	0.000896541	0.0223332
30	0.3	5.3816	0.000991527	0.029612
35	0.35	5.92537	0.00109663	0.0381733
40	0.4	6.52329	0.00121291	0.0482067
45	0.45	7.18104	0.00134154	0.059927
50	0.5	7.9049	0.00148381	0.073578
55	0.55	8.70181	0.00164117	0.0894361
60	0.6	9.57944	0.00181519	0.107814
65	0.65	10.5463	0.00200761	0.129067
70	0.7	11.6116	0.00222039	0.153594
75	0.75	12.7859	0.00245564	0.18185
80	0.8	14.0804	0.00271572	0.214347
85	0.85	15.5079	0.00300326	0.251663
90	0.9	17.0822	0.00332112	0.294452
95	0.95	18.8188	0.00367249	0.343449
100	1.	20.7345	0.0040609	0.399488

+ Definitions

Butcher tableau:

$$\begin{array}{c|c} 1 & \\ \hline & 1 \end{array}$$

Symbolic iteration code:

$$y'(t) = f(t, y) = 2\,y(t) - \sin(t),\ y(0) = 3$$

$$y_{n+1} = y_n + k_1$$

$$t_{n+1} = t_n + h$$

$$k_1 = h\,f(t_n, y_n)$$

$$k_2 = h\,f(t_n + h, y_n + \ h\,k_1)$$

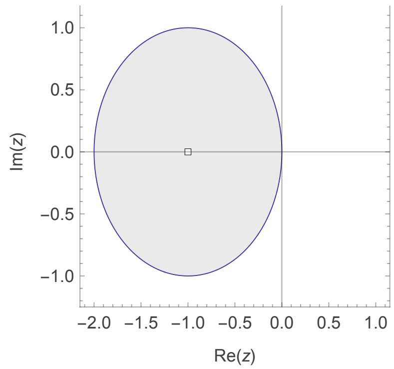
where $y_0 = 3$

$t_0 = 0$

$h = 0.01$

$n = 0,\ldots,100$

Stability region in complex stepsize plane:

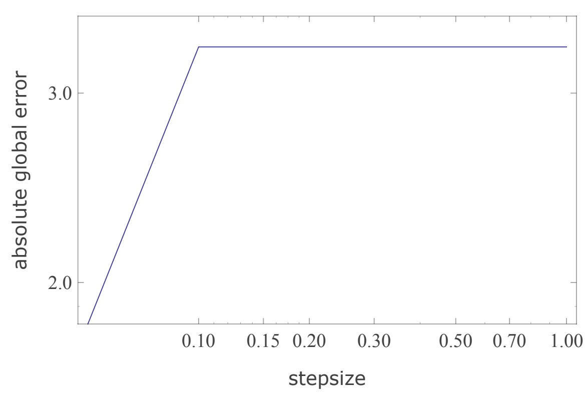


stability function: $z + 1$

Exact solution of equation:

$$y(t) = \frac{1}{5}\left(14\, e^{2t} + 2\sin(t) + \cos(t)\right)$$

Stepsize comparison:



(global error at t == 1)

Method comparison:

method	global error	log scale comparison
forward Euler method	0.399	
midpoint method	0.00269	
Heun's method	0.00268	
third- order Runge- Kutta method	0.0000134	
fourth- order Runge- Kutta method	5.36×10^{-8}	
Runge- Kutta- Fehlberg method	-2.16×10^{-6}	
Bogacki- Shampine method	5.72×10^{-7}	
Dormand- Prince method	-6.39×10^{-8}	

backward Euler method	−0.419	
implicit midpoint method	−0.00136	

(global error at $t = 1$)

Mathematica input:

```
NDSolve[{y'[t] == -Sin[t] + 2 y[t], y[0] == 3}, y,
  {t, 0, 1}, Method-> {"FixedStep", Method -> "ExplicitEuler"},
   StartingStepSize -> 0.01, WorkingPrecision -> MachinePrecision]
```

WolframAlpha