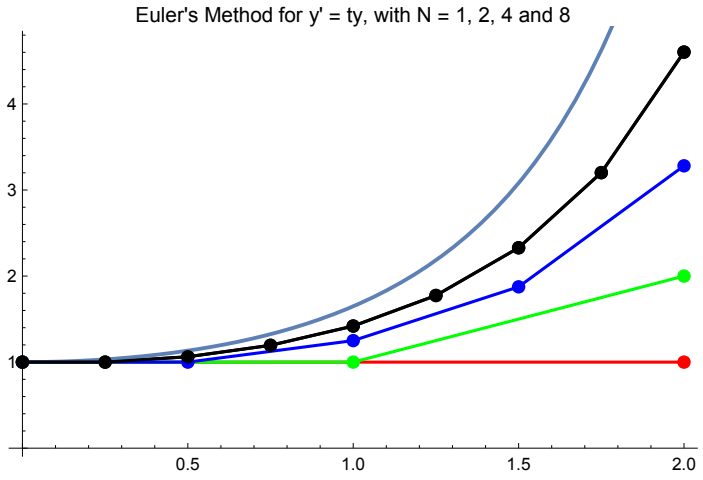
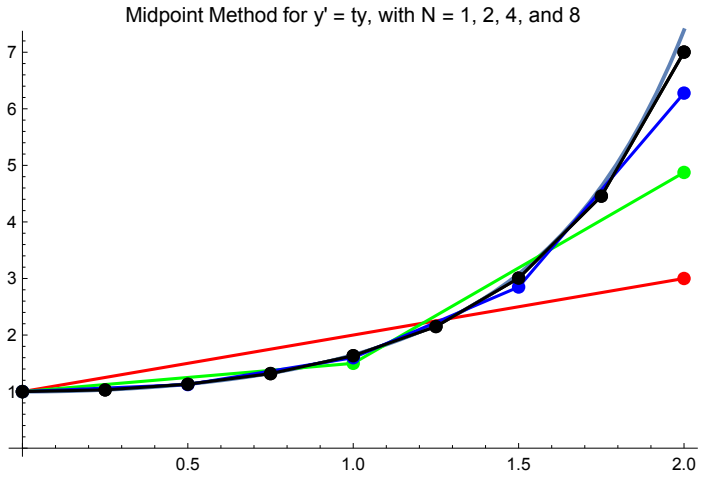
In[409]:=
```
f[t_, y_] :=  t y;
ytrue[t_] := Exp[t^2 / 2];
a = 0;
b = 2;
Nits = 1;
y0 = 1;
p0 = Plot[ytrue[t], {t, a, b}, PlotStyle → Thick];
p1 = myListPlot[euler[a, b, Nits, y0], Red];
p2 = myListPlot[euler[a, b, 2 * Nits, y0], Green];
p3 = myListPlot[euler[a, b, 4 * Nits, y0], Blue];
p4 = myListPlot[euler[a, b, 8 * Nits, y0], Black];
Show[p4, p0, p1, p2, p3, p4,
 PlotLabel → "Euler's Method for y' = ty, with N = 1, 2, 4 and 8"]
```

Out[420]=



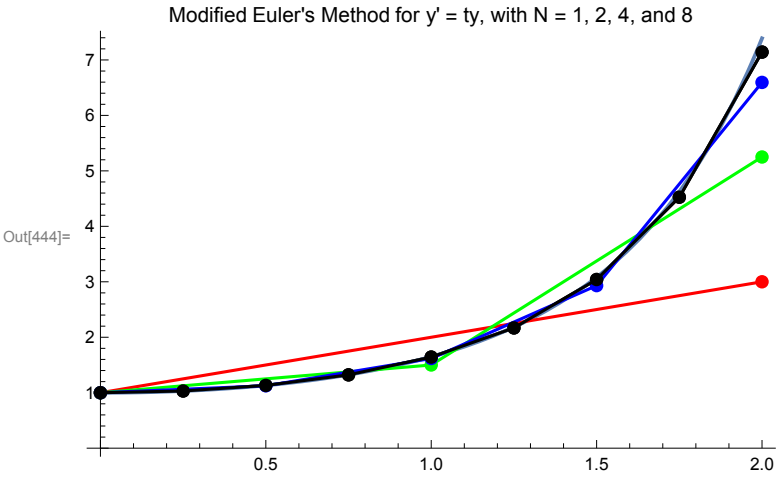
In[421]:=
```
f[t_, y_] := t y;
ytrue[t_] := Exp[t^2 / 2];
a = 0;
b = 2;
Nits = 1;
y0 = 1;
p0 = Plot[ytrue[t], {t, a, b}, PlotStyle → Thick];
p1 = myListPlot[midpoint[a, b, Nits, y0], Red];
p2 = myListPlot[midpoint[a, b, 2 * Nits, y0], Green];
p3 = myListPlot[midpoint[a, b, 4 * Nits, y0], Blue];
p4 = myListPlot[midpoint[a, b, 8 * Nits, y0], Black];
Show[p4, p0, p1, p2, p3, p4,
 PlotLabel → "Midpoint Method for y' = ty, with N = 1, 2, 4, and 8"]
```

Out[432]=



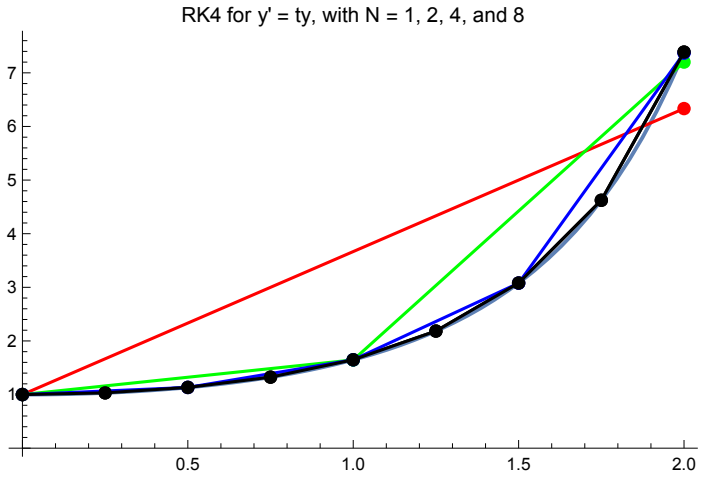
In[433]:=
```
f[t_, y_] := t y;
ytrue[t_] := Exp[t^2 / 2];
a = 0;
b = 2;
Nits = 1;
y0 = 1;
p0 = Plot[ytrue[t], {t, a, b}, PlotStyle → Thick];
p1 = myListPlot[modifiedEuler[a, b, Nits, y0], Red];
p2 = myListPlot[modifiedEuler[a, b, 2 * Nits, y0], Green];
p3 = myListPlot[modifiedEuler[a, b, 4 * Nits, y0], Blue];
p4 = myListPlot[modifiedEuler[a, b, 8 * Nits, y0], Black];
Show[p4, p0, p1, p2, p3, p4,
 PlotLabel → "Modified Euler's Method for y' = ty, with N = 1, 2, 4, and 8"]
```

Out[444]=



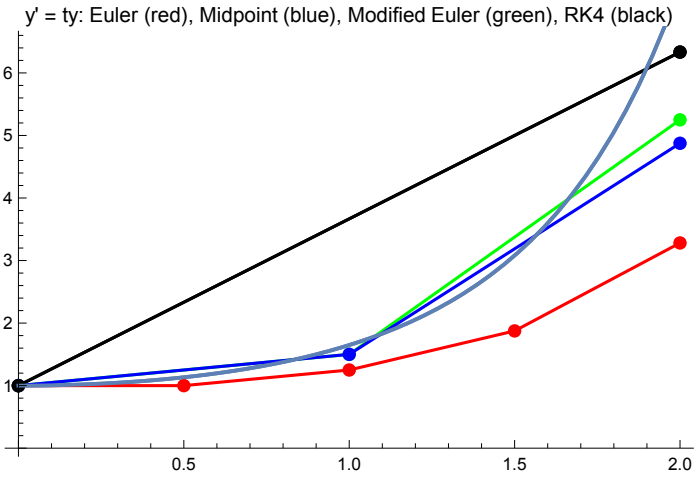
In[445]:=
```
f[t_, y_] := t y;
ytrue[t_] := Exp[t^2 / 2];
a = 0;
b = 2;
Nits = 1;
y0 = 1;
p0 = Plot[ytrue[t], {t, a, b}, PlotStyle → Thick];
p1 = myListPlot[rk4[a, b, Nits, y0], Red];
p2 = myListPlot[rk4[a, b, 2 * Nits, y0], Green];
p3 = myListPlot[rk4[a, b, 4 * Nits, y0], Blue];
p4 = myListPlot[rk4[a, b, 8 * Nits, y0], Black];
Show[p4, p0, p1, p2, p3, p4,
 PlotLabel → "RK4 for y' = ty, with N = 1, 2, 4, and 8"]
```

Out[456]=



In[457]:=
```
f[t_, y_] := t y;
ytrue[t_] := Exp[t^2 / 2];
a = 0;
b = 2;
Nits = 1;
y0 = 1;
p0 = Plot[ytrue[t], {t, a, b}, PlotStyle → Thick];
p1 = myListPlot[euler[a, b, 4 * Nits, y0], Red];
p2 = myListPlot[modifiedEuler[a, b, 2 * Nits, y0], Green];
p3 = myListPlot[midpoint[a, b, 2 * Nits, y0], Blue];
p4 = myListPlot[rk4[a, b, Nits, y0], Black];
Show[p4, p0, p1, p2, p3, p4, p0, PlotLabel →
  "y' = ty: Euler (red), Midpoint (blue), Modified Euler (green), RK4 (black)"]
```

Out[468]=



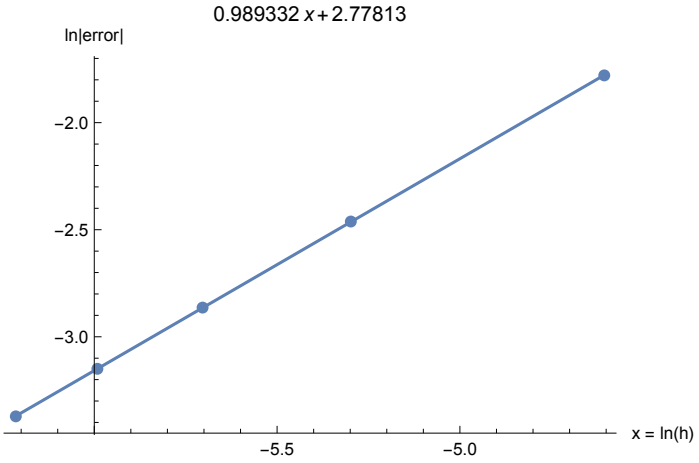
This figure is a “fair comparison” of the different methods, because each requires 4 evaluations of f(t, y).

In[469]:=
```
f[t_, y_] := t y;
ytrue[t_] := Exp[t^2 / 2];
a = 0;
b = 2;
y0 = 1;
Clear[t, y];
Print["Euler approximations to y' = ",
  f[t, y], ", ", a , " ≤ t ≤ ", b, ", y(0) = 1" ];
Nmin = 200;
Nmax = 1000;
Nstep = 200;
eulerTable = {};
loglog = {};
For[Nits = Nmin, Nits ≤ Nmax, Nits += Nstep,
 y = y0;
 h = (b - a) / Nits;
 soln = euler[a, b, Nits, y];
 myListPlot[soln, Red];
 lastPoint = Part[soln, -1]; (* The last (t, y) point*)
 (*Print[lastPoint];*)(* comment this out after checking *)
 lasty = Part[lastPoint, 2]; (* Take last y value *)
 error = ytrue[b] - lasty;
 AppendTo[eulerTable, {Nits, lasty, error, error * Nits}];
 AppendTo[loglog, {Log[h], Log[Abs[error]]}];
]
Print@TableForm[eulerTable, TableHeadings →
    {None, {"N", "approx of y(2)", "error of approx", "error*N"}}];
p1 = ListPlot[loglog];
line = Fit[loglog, {1, x}, x];
p2 = Plot[line, {x, Log[(b - a) / Nmax], Log[(b - a) / Nmin]},
   PlotLabel → line, AxesLabel → {"x = ln(h)", "ln|error|"}];
Show[p2, p1]
```

Euler approximations to y' = t y, 0 ≤ t ≤ 2, y(0) = 1

N	approx of y(2)	error of approx	error*N
200	7.2203	0.168753	33.7506
400	7.30377	0.0852815	34.1126
600	7.332	0.0570583	34.235
800	7.34619	0.0428706	34.2965
1000	7.35472	0.0343335	34.3335

Out[486]=



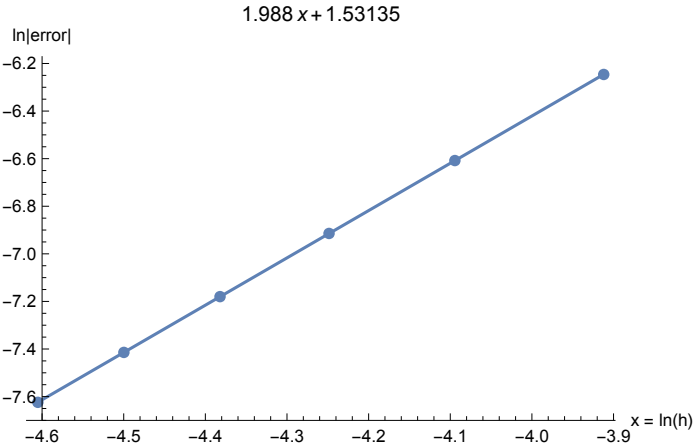
The table and graph compute error of the final approximation to y as a function of N, or h =(b-a)/N, for the Euler Method. The expected O(h) error is demonstrated. The slope of a Log[error] vs. Log[h] graph is very close to 1, especially when N gets large.

In[487]:=
```
f[t_, y_] := t y;
ytrue[t_] := Exp[t^2 / 2];
a = 0;
b = 2;
y0 = 1;
Clear[t, y];
Print["Modified Euler approximations to y' = ",
  f[t, y], ", ", a , " ≤ t ≤ ", b , ", y(0) = 1"];
Nmin = 100;
Nmax = 200;
Nstep = 20;
modifiedEulerTable = {};
loglog = {};
For[Nits = Nmin, Nits ≤ Nmax, Nits += Nstep,
 y = y0;
 h = (b - a) / Nits;
 soln = modifiedEuler[a, b, Nits, y];
 myListPlot[soln, Red];
 lastPoint = Part[soln, -1]; (* The last (t, y) point*)
 (*Print[lastPoint];*)(* comment this out after checking *)
 lasty = Part[lastPoint, 2]; (* Take last y value *)
 error = ytrue[b] - lasty;
 AppendTo[modifiedEulerTable, {Nits, lasty, error, error * Nits^2}];
 AppendTo[loglog, {Log[h], Log[Abs[error]]}];
]
Print@TableForm[modifiedEulerTable, TableHeadings →
    {None, {"N", "approx of y(2)", "error of approx", "error*N^2"}}];
p1 = ListPlot[loglog];
line = Fit[loglog, {1, x}, x];
p2 = Plot[line, {x, Log[(b - a) / Nmax], Log[(b - a) / Nmin]},
   PlotLabel → line, AxesLabel → {"x = ln(h)", "ln|error|"}];
Show[p2, p1]
```

Modified Euler approximations to y' = t y, 0 ≤ t ≤ 2, y(0) = 1

N	approx of y(2)	error of approx	error*N^2
100	7.38712	0.00193789	19.3789
120	7.38771	0.00134952	19.4331
140	7.38806	0.000993462	19.4718
160	7.38829	0.000761753	19.5009
180	7.38845	0.000602576	19.5235
200	7.38857	0.000488539	19.5415

Out[504]=



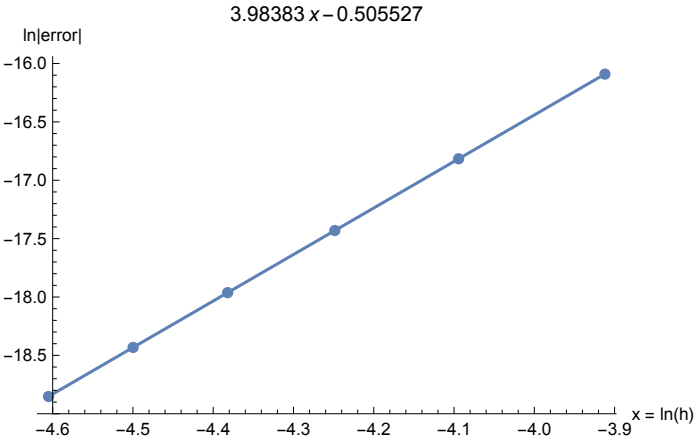
The table and graph compute error as a function of N, or h =(b-a)/N, for the Modified Euler Method. The expected O(h^2) error is demonstrated. The slope of a Log[error] vs. Log[h] graph is close to 2, especially when N gets large. The Midpoint method results are similar, and they are not shown here.

In[505]:=
```
f[t_, y_] := t y;
ytrue[t_] := Exp[t^2 / 2];
a = 0;
b = 2;
y0 = 1;
Clear[t, y];
Print["RK4 approximations to y' = ",
  f[t, y], ", ", a , " ≤ t ≤ ", b, ", y(0) = 1" ];
Nmin = 100;
Nmax = 200;
Nstep = 20;
rk4Table = {};
loglog = {};
For[Nits = Nmin, Nits ≤ Nmax, Nits += Nstep,
 y = y0;
 h = (b - a) / Nits;
 soln = rk4[a, b, Nits, y];
 myListPlot[soln, Red];
 lastPoint = Part[soln, -1]; (* The last (t, y) point*)
 (*Print[lastPoint];*)(* comment this out after checking *)
 lasty = Part[lastPoint, 2]; (* Take last y value *)
 error = ytrue[b] - lasty;
 AppendTo[rk4Table, {Nits, lasty, error, error * Nits^4}];
 AppendTo[loglog, {Log[h], Log[Abs[error]]}];
]
Print@TableForm[rk4Table, TableHeadings →
    {None, {"N", "approx of y(2)", "error of approx", "error*N^4"}}];
p1 = ListPlot[loglog];
line = Fit[loglog, {1, x}, x];
p2 = Plot[line, {x, Log[(b - a) / Nmax], Log[(b - a) / Nmin]},
   PlotLabel → line, AxesLabel → {"x = ln(h)", "ln|error|"}];
Show[p2, p1]
```

RK4 approximations to y' = t y, 0 ≤ t ≤ 2, y(0) = 1

N	approx of y(2)	error of approx	error*N^4
100	7.38906	1.02756×10^{-7}	10.2756
120	7.38906	4.97409×10^{-8}	10.3143
140	7.38906	2.6921×10^{-8}	10.342
160	7.38906	1.58123×10^{-8}	10.3628
180	7.38906	9.88697×10^{-9}	10.3789
200	7.38906	6.49494×10^{-9}	10.3919

Out[522]=



The table and graph compute error as a function of N, or h =(b-a)/N, for the Runge-Kutta 4 Method. The expected O(h^4) error is demonstrated. The slope of a Log[error] vs. Log[h] graph is close to 4, especially when N gets large.