```
> restart:
  with(plots):
  Digits = 14:
```

# Absolute stability of various one-step stencils

In this worksheet, we would like to develop an intuitive understanding of what it means for a stencil to be "absolutely stable" by conducting some numerical experiments, and to determine under which conditions one-step stecils for solving $y'(x) = f(x, y(x))$ are absolutely stable.

## Some numerical experiments

Here are five separate one-step stencils for solving $\frac{\mathrm{d}}{\mathrm{d}x} y(x) = f(x, y(x))$:

```
> Stencil[1] := y_new - (y_old+f(x_old,y_old)*h)=0:
  Stencil[2] := y_new - y_old - f(x_new,y_new)*h = 0:
  Stencil[3] := y_new-y_old-1/2*f(x_old,y_old)*h-1/2*f(x_new,y_new)*h = 0:
  Stencil[4] := y_new - (y_old+h*(1/2*f(x_old,y_old)+1/2*f(x_old+h,y_old+f(x_old,y_old)*h)))
  =0:
  Stencil[5] := y_new - (y_old+h*(1/6*f(x_old,y_old)+1/3*f(x_old+1/2*h,y_old+1/2*h*f(x_old,
  y_old))+1/3*f(x_old+1/2*h,y_old+1/2*h*f(x_old+1/2*h,y_old+1/2*h*f(x_old,y_old))+1/6*f
  (x_old+h,y_old+h*f(x_old+1/2*h,y_old+1/2*h*f(x_old+1/2*h,y_old+1/2*h*f(x_old,y_old))))))=
  0:

  stencil := convert(Stencil,list):

  Labels := [`Forward Euler`,`Backward Euler`,`Trapezoidal`,`Huen`,`RK4`]:

  for i from 1 to 5 do:
      Labels[i],stencil[i]:
  od;
```

$$Forward\ Euler,\ y\_new - y\_old - f(x\_old, y\_old)\ h = 0$$

$$Backward\ Euler,\ y\_new - y\_old - f(x\_new, y\_new)\ h = 0$$

$$Trapezoidal,\ y\_new - y\_old - \frac{1}{2} f(x\_old, y\_old)\ h - \frac{1}{2} f(x\_new, y\_new)\ h = 0$$

$$Huen,\ y\_new - y\_old - h\left(\frac{1}{2} f(x\_old, y\_old) + \frac{1}{2} f(x\_old + h, y\_old + f(x\_old, y\_old)\ h)\right) = 0$$

$$RK4, y\_new - y\_old - h\left(\frac{1}{6}f(x\_old, y\_old) + \frac{1}{3}f\left(x\_old + \frac{1}{2}h, y\_old + \frac{1}{2}f(x\_old, y\_old)\,h\right) + \frac{1}{3}f\left(x\_old + \frac{1}{2}h, y\_old + \frac{1}{2}hf\left(x\_old + \frac{1}{2}h, y\_old + \frac{1}{2}f(x\_old, y\_old)\,h\right)\right) + \frac{1}{6}f\left(x\_old + h, y\_old + hf\left(x\_old + \frac{1}{2}h, y\_old + \frac{1}{2}hf\left(x\_old + \frac{1}{2}h, y\_old + \frac{1}{2}f(x\_old, y\_old)\,h\right)\right)\right)\right) = 0 \quad \textbf{(1.1}$$

We're going to initially limit ourselves to the choice $f(x, y) = \lambda y$, where $\lambda$ is meant to be an arbitrary complex number. Also, let's fix the initial data to be $y(0) = 1$. There is hence a simple analytic solution which will correspond to the Maple mapping **y_sol**:

```
> f := (x,y) -> lambda*y;
  ode := diff(y(x),x) = f(x,y(x));
  IC := y(0) = 1;
  sol := dsolve({ode,IC});
  y_sol := unapply(rhs(sol),x);
```

$$f := (x, y) \to \lambda y$$

$$ode := \frac{d}{dx}y(x) = \lambda y(x)$$

$$IC := y(0) = 1$$

$$sol := y(x) = e^{\lambda x}$$

$$y\_sol := x \to e^{\lambda x} \quad \textbf{(1.2}$$

Under these circumstances, each of the above stencils can be solved for **y_new**:

```
> stencil := map(x->factor(simplify(isolate(x,y_new))),stencil);
```

$$stencil := \left[y\_new = y\_old\,(1 + \lambda h), y\_new = -\frac{y\_old}{-1 + \lambda h}, y\_new = -\frac{y\_old\,(2 + \lambda h)}{-2 + \lambda h}, y\_new = \frac{1}{2}y\_old\,(2 + 2\lambda h + \lambda^2 h^2), y\_new = \frac{1}{24}y\_old\,(24 + 24\lambda h + 12\lambda^2 h^2 + 4\lambda^3 h^3 + \lambda^4 h^4)\right] \quad \textbf{(1.3}$$

We want to convert these to numerical algorithms, so we need to write these as mappings:

```
> stencil := map(x->unapply(rhs(x),h,lambda,y_old),stencil):
  for i from 1 to 5 do:
       Labels[i],stencil[i];
  od;
```

$$Forward\ Euler, (h, \lambda, y\_old) \to y\_old\,(1 + \lambda h)$$

$$\text{Backward Euler}, (h, \lambda, y\_old) \to -\frac{y\_old}{-1 + \lambda h}$$

$$\text{Trapezoidal}, (h, \lambda, y\_old) \to -\frac{y\_old\,(2 + \lambda h)}{-2 + \lambda h}$$

$$\text{Huen}, (h, \lambda, y\_old) \to \frac{1}{2}\, y\_old\,(2 + 2\lambda h + \lambda^2 h^2)$$

$$\text{RK4}, (h, \lambda, y\_old) \to \frac{1}{24}\, y\_old\,(24 + 24\lambda h + 12\lambda^2 h^2 + 4\lambda^3 h^3 + \lambda^4 h^4) \qquad \textbf{(1.4}$$

Observe that while we have written each stencil as a function of $h$ and $\lambda$ separately, they are really functions of $z \equiv \lambda h$. (From this it follows that all the stability properties of this stencil will depend on the value of $z$.) Now, here is some code that generates the numeric solution to $y' = \lambda y$ on the interval $x \in [0, X]$ subject to initial conditions $y(1) = 1$ and using a stepsize of $h$. The parameter **choice** indicates which stencil to use (1 = forward Euler, 2 = backward Euler, etc.):

```
> Sol := proc(lambda,h,X,choice)
      local N, x, y, i:

      N := round(X/h)+1:
      x[0] := 0:
      y[0] := 1:
      for i from 1 to N do:
         x[i] := x[i-1] + h:
         y[i] := stencil[choice](h,lambda,y[i-1]):
      od:
      [seq([x[i],y[i]],i=0..N)]:

  end proc:
```

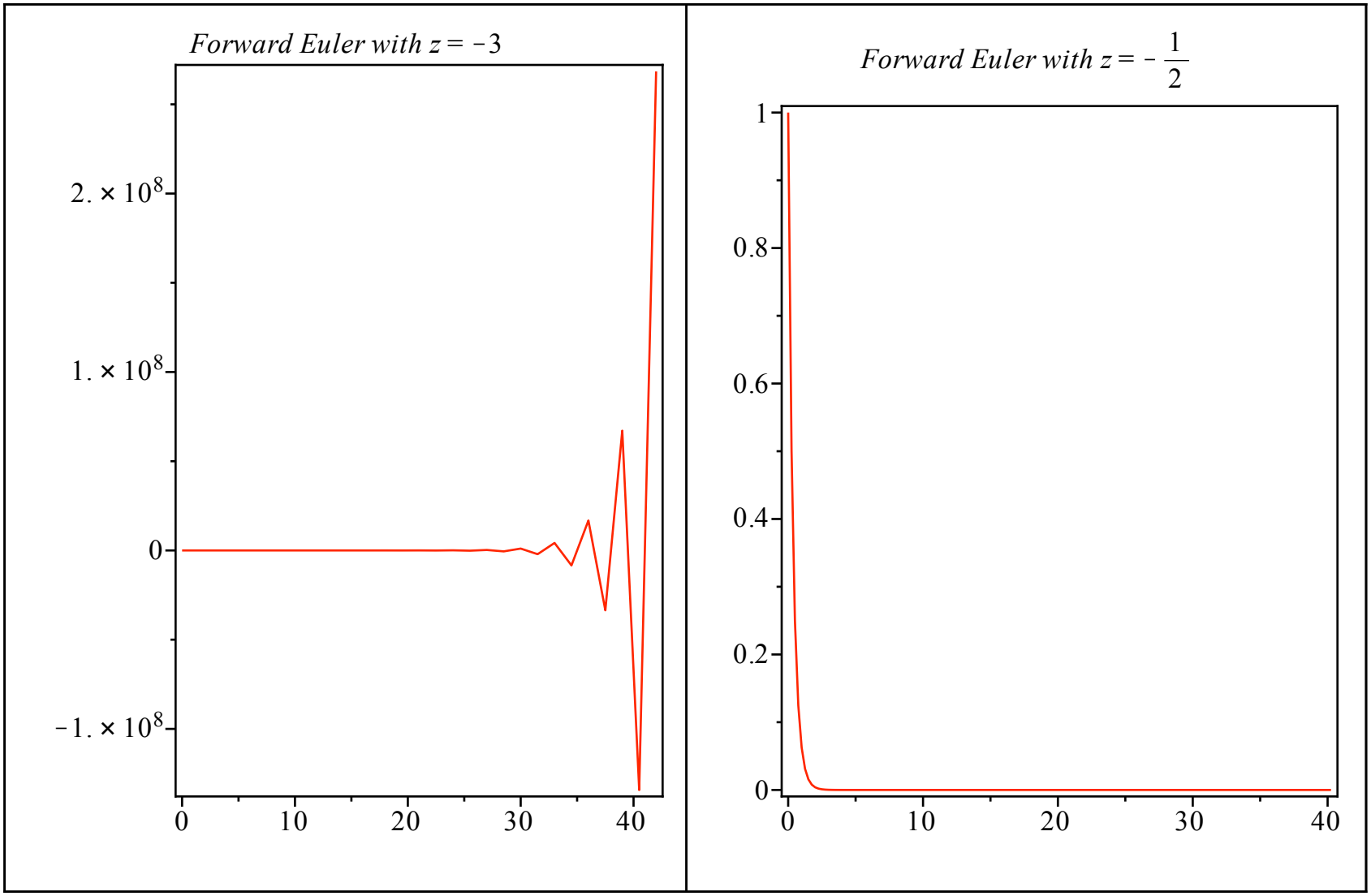
The procedure **plotter1** creates a plot of the numeric solution for a given stencil and other choices of parameters. It includes information about the value of $z$ (which can be thought of as the stepsize in units of $\lambda^{-1}$) for each simulation in the plot's title:

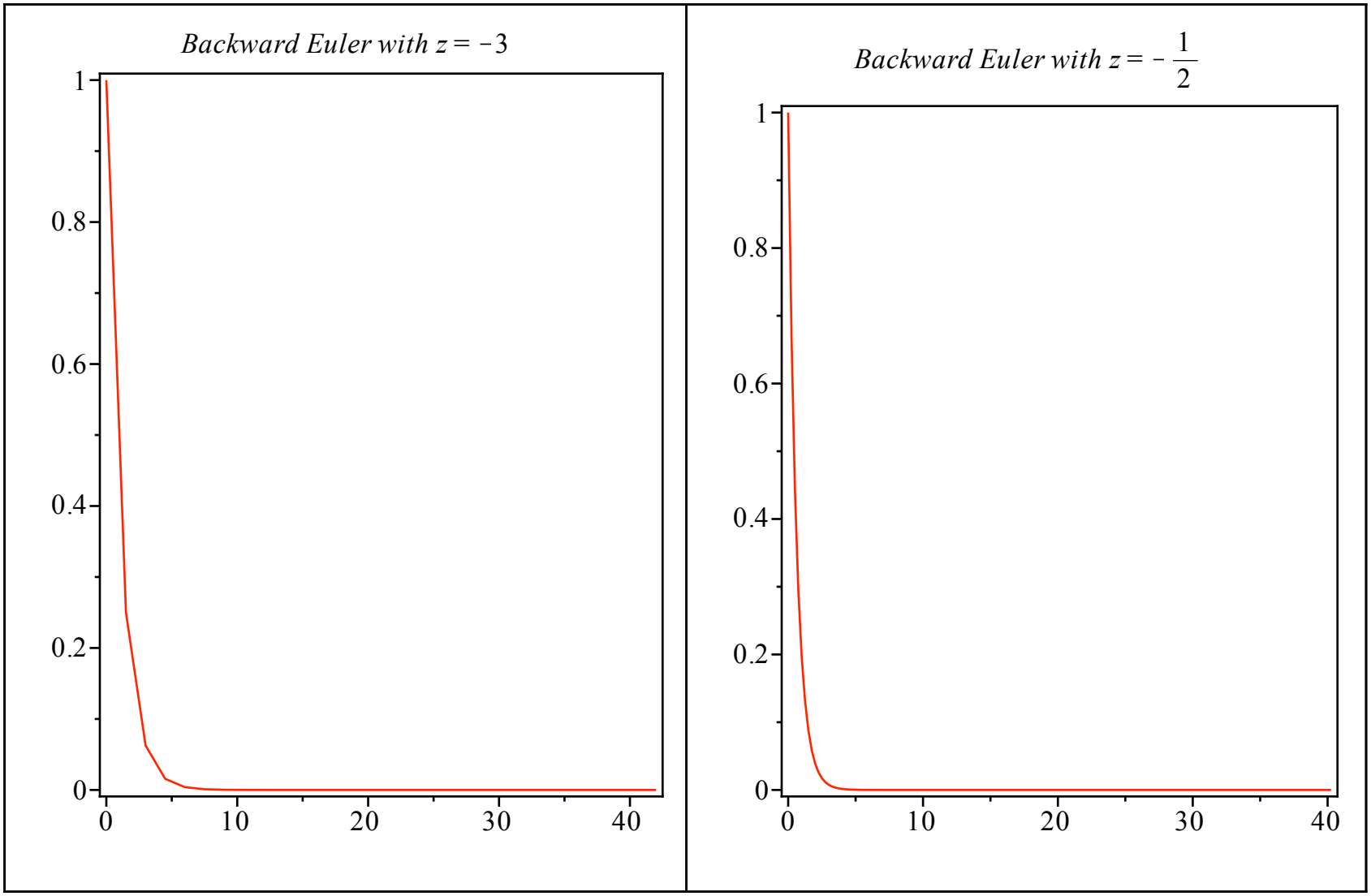
```
> plotter1 := (lambda,h,X,choice) -> plot(Sol(lambda,h,X,choice),axes=boxed,title=typeset
  (cat(Labels[choice],` with `),z=lambda*h));
```

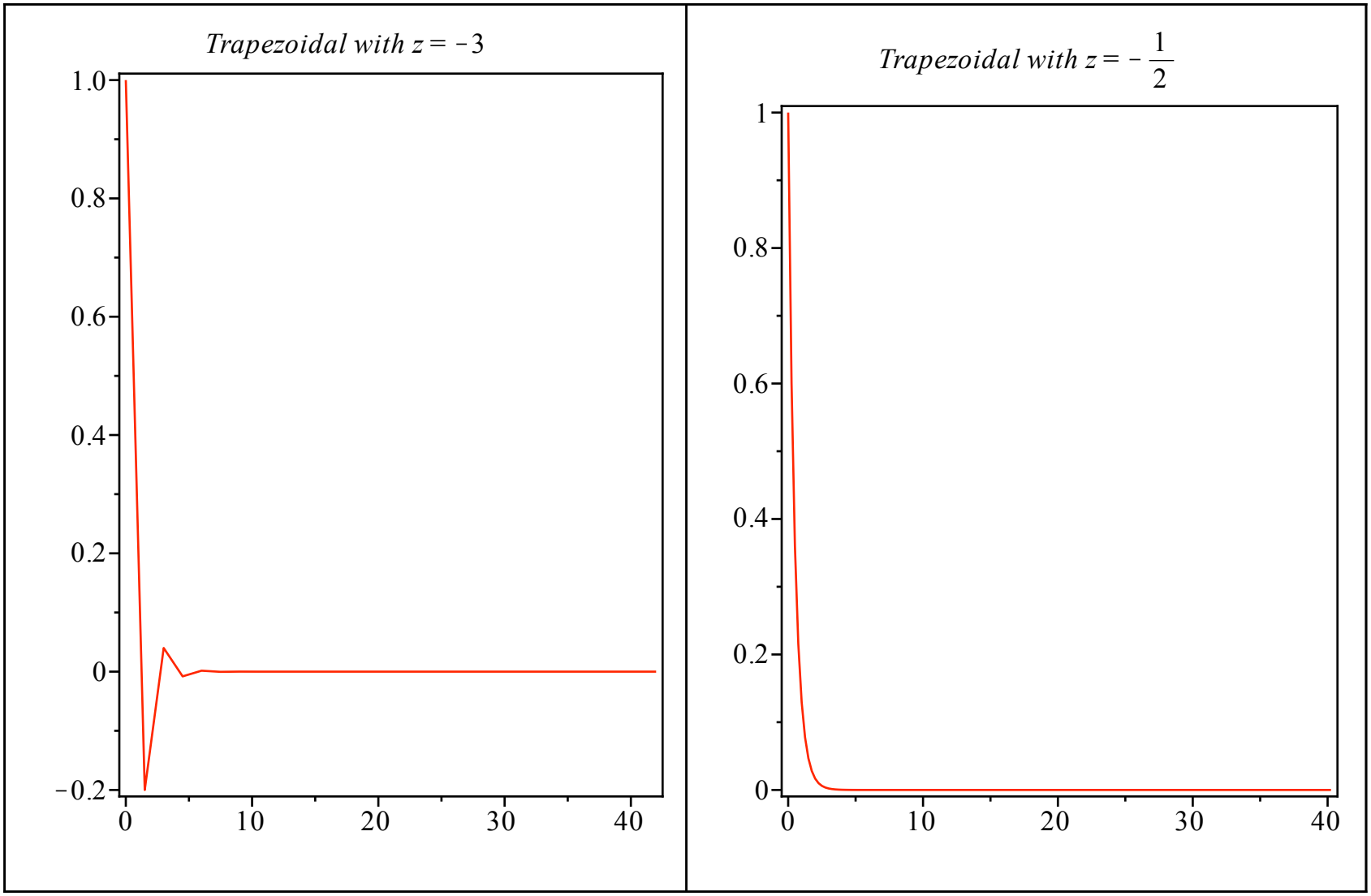
$$plotter1 := (\lambda, h, X, choice) \to plot\left(Sol(\lambda, h, X, choice), axes = boxed, title = typeset\left(cat\left(Labels_{choice}, \text{ with }\right), z = h\lambda\right)\right) \qquad \textbf{(1.5}$$

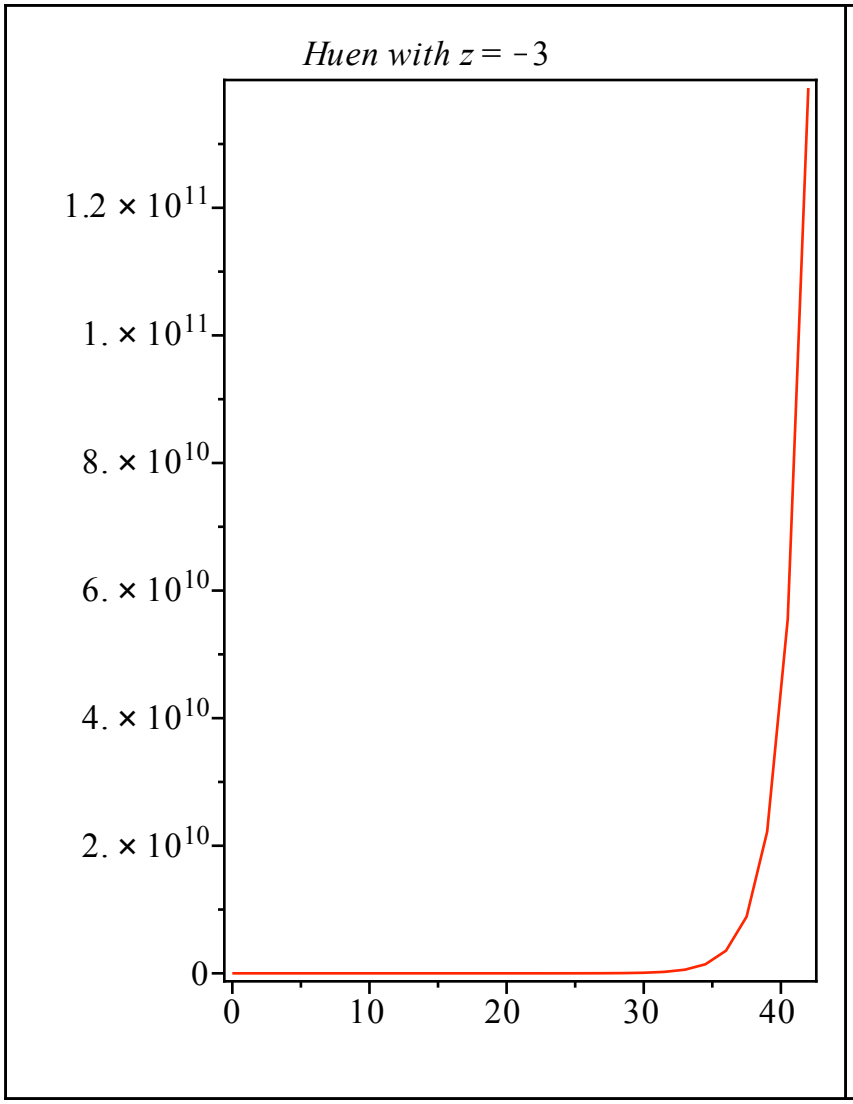
We now create a pair of plots of the solution for each stencil (all of these plots have $\lambda = -2$ and $X = 40$). For the first member of each pair we choose the stepsize such that $z = -3$; for the second $z = -\frac{1}{2}$. Note how we use the **display** command to arrange the plots in a panel:

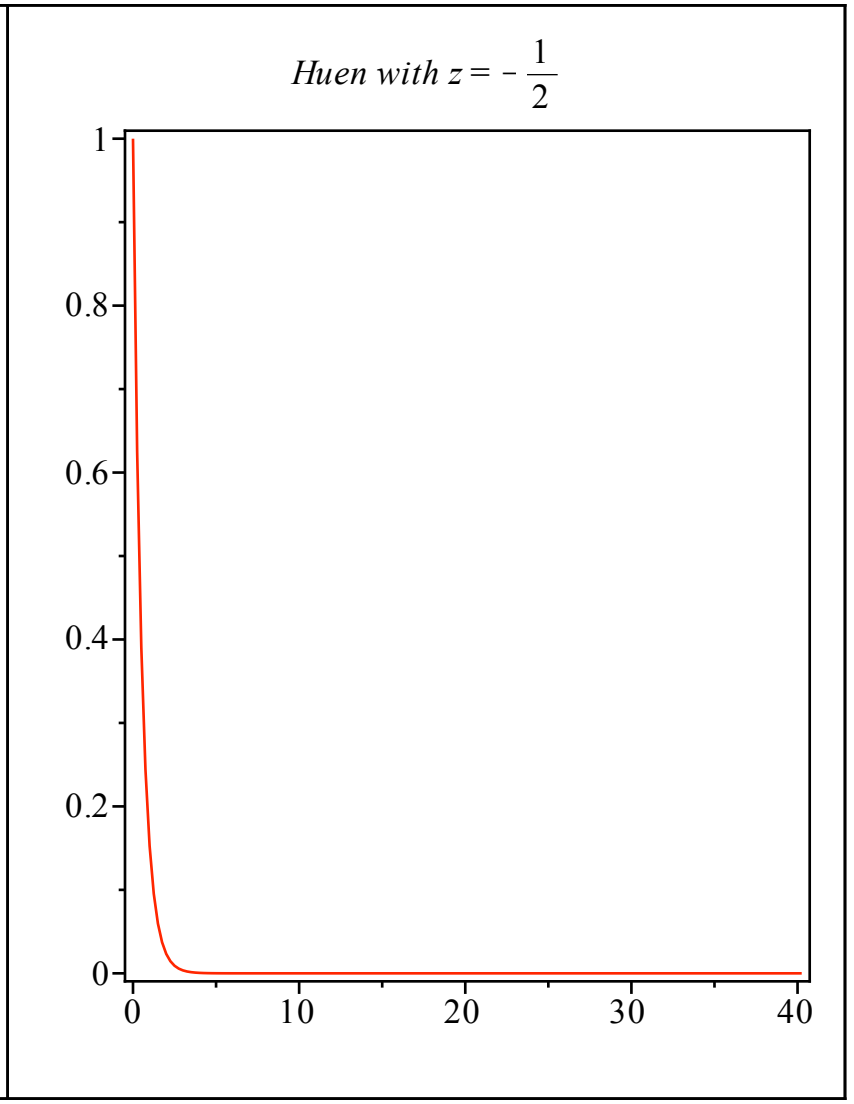
```
> p := Array(1..5,1..2):

  H[1] := 3/2:
  H[2] := 1/4:

  lambda := -2:
  X := 40:
  for i from 1 to 5 do:
    for j from 1 to 2 do:
      p[i,j] := plotter1(lambda,H[j],X,i);
    od:
  od:

  display(p);
```

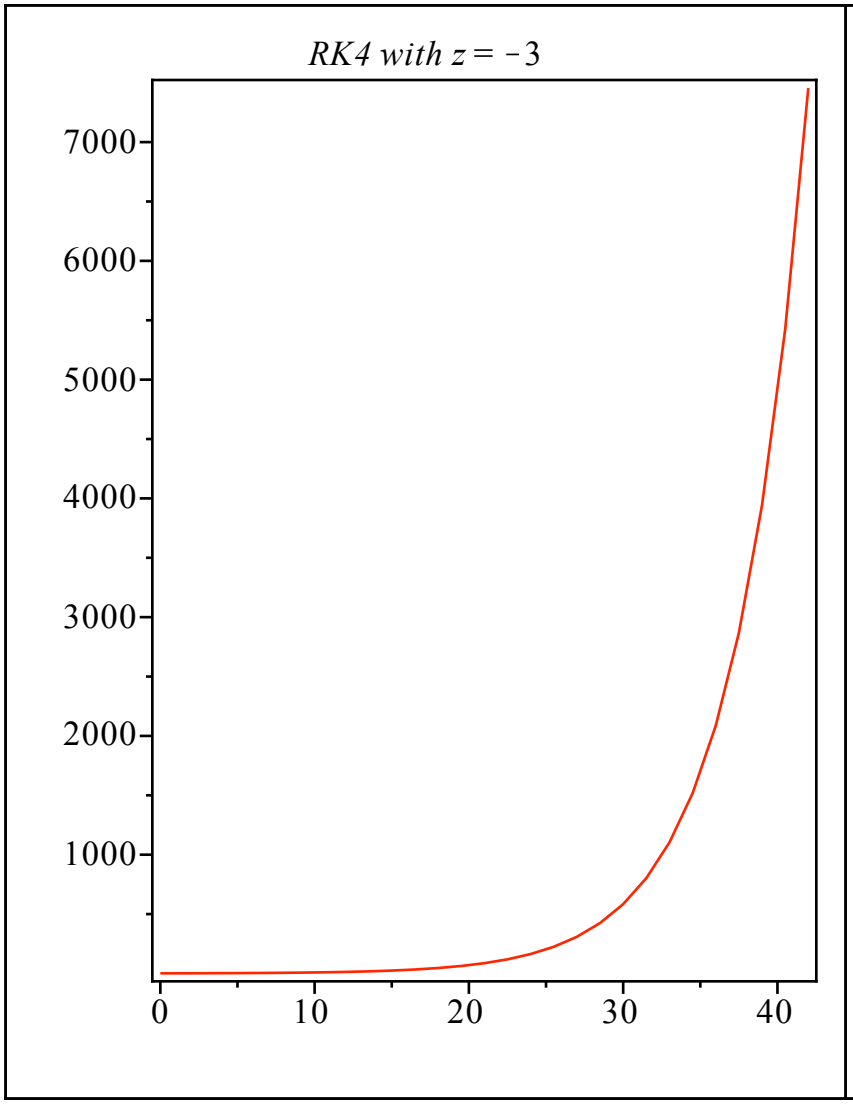

Forward Euler with $z = -3$
$2. \times 10^8$
$1. \times 10^8$
0
$-1. \times 10^8$
0
10
20
30
40
Forward Euler with $z = -\frac{1}{2}$
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

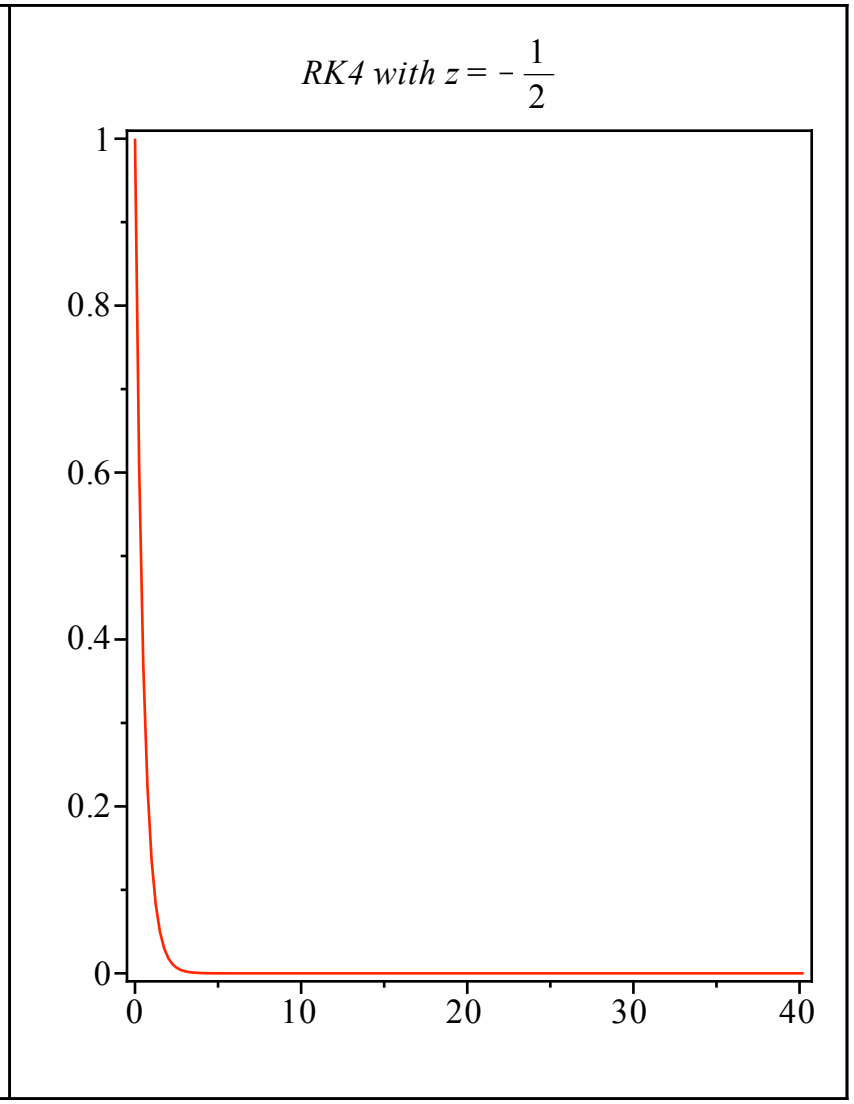

Backward Euler with $z = -3$
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
Backward Euler with $z = -\frac{1}{2}$
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40


Trapezoidal with z = -3
1.0
0.8
0.6
0.4
0.2
0
-0.2
0
10
20
30
40
Trapezoidal with z = - 1/2
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40


Huen with z = −3
1.2 × 10^11
1. × 10^11
8. × 10^10
6. × 10^10
4. × 10^10
2. × 10^10
0
0
10
20
30
40


Huen with z = −1/2
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40




The exact solution for these parameters is $y(x) = e^{-2x}$. We see that all the stencil's output *qualitatively* resembles this decaying exponential for $z = -\frac{1}{2}$. The backward Euler and trapezoidal stencils also capture the decay of the solution for $z = -3$ (though the trapezoidal results involve some suprious oscillations). But the forward Euler, Huen, and 4th order Runge-Kutta completely miss the basic shape of the exact solution; the numeric results appear to be exponentially growing in magnitude, not decaying. Since the absolute discrepancy between the

exact and numerical result blows up as $x$ increases, we say that these stencils are *unstable* for $z = -3$ (in an absolute sense). We have the following definition:

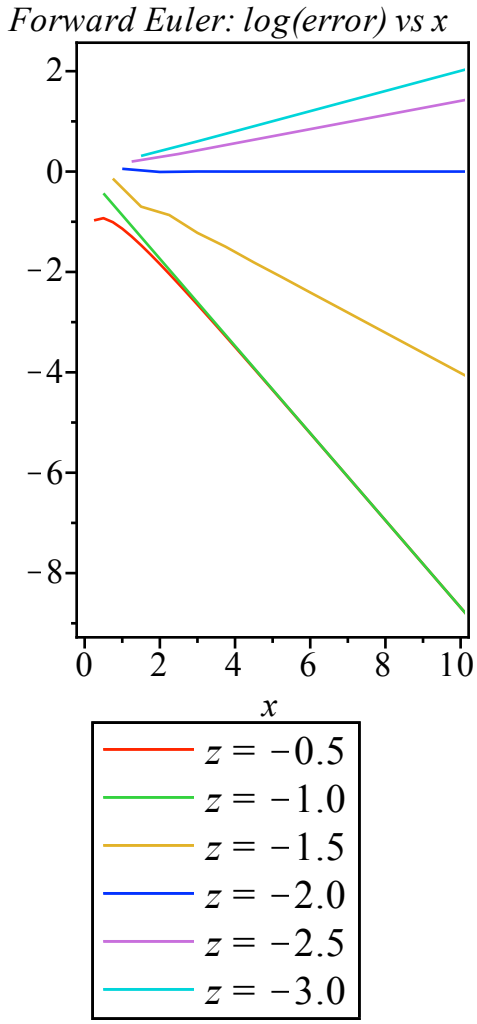
**Definition:** A numerical stencil is said to be *absolutely stable* if the discrepancy between the numerical approximation and the exact solution does not grow exponentially as $x \to \infty$.

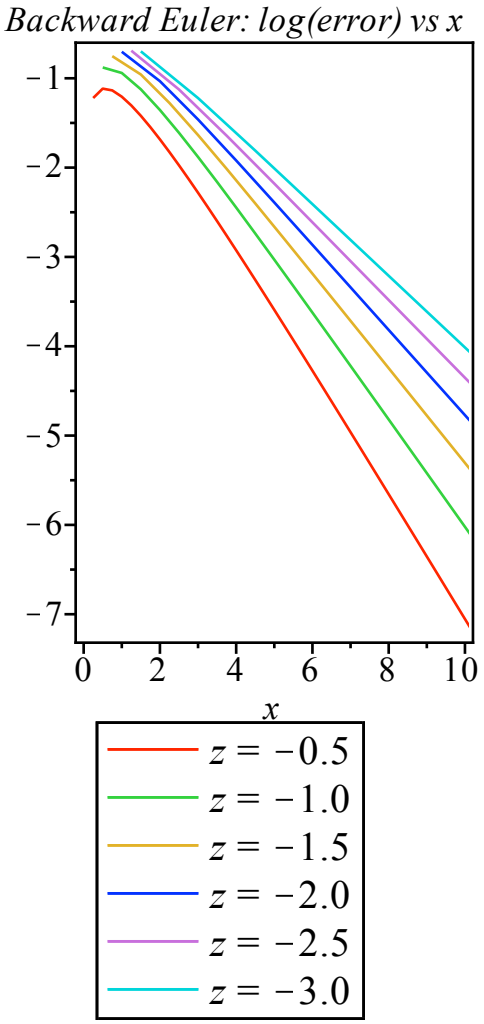
We can get a more systematic view of what's going on by plotting the global error (i.e. the absolute value of the difference between the exact and numeric solutions) as a function of $x$ for various values of $z$. The following procedure generates the data for this type of plot (with the errors reported as a logarithm):

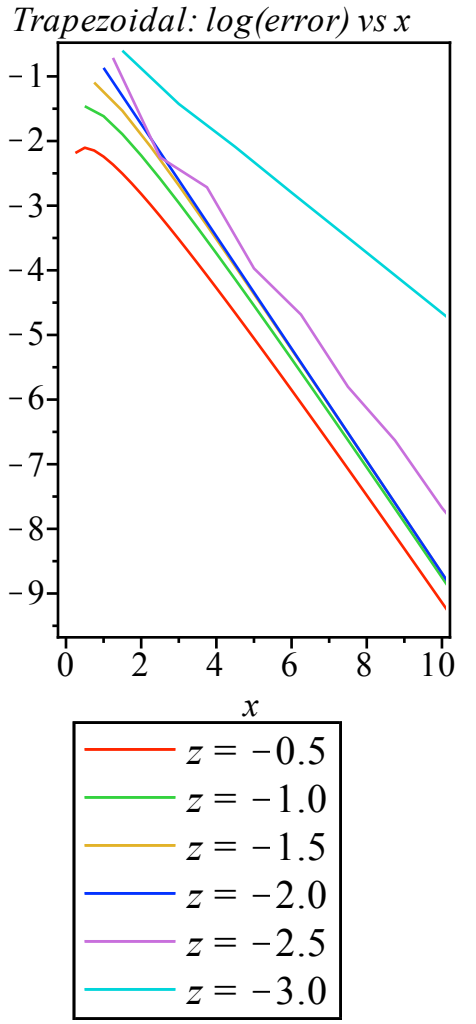
```
> LogError := proc(lambda,z,X,choice) local data, N;
      data := Sol(lambda,abs(z/lambda),X,choice):
      N := nops(data):
      data := map(x->[x[1],log10(abs(x[2]-exp(lambda*x[1])))],data[2..N]);
      data := evalf(data);
  end proc:
```

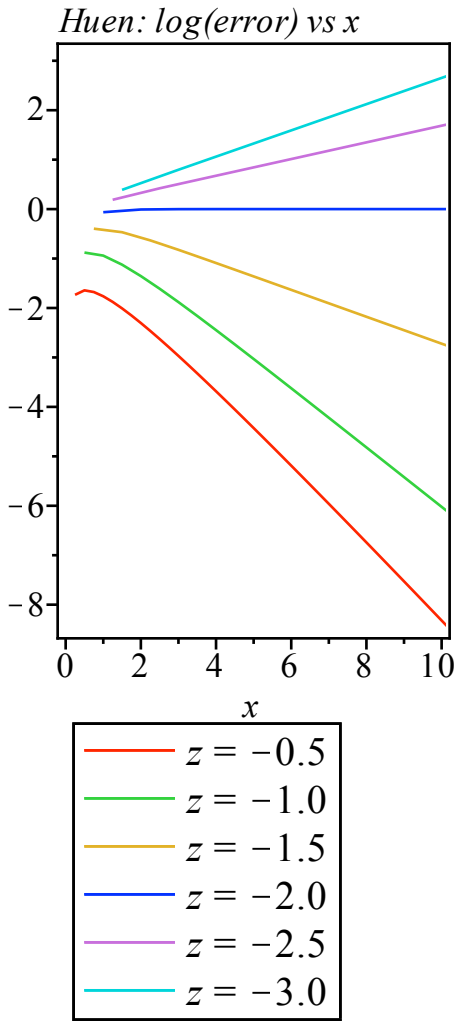
Here are the plots of the global errors as a function of $x$. We see that the two implicit stencils have global errors that are bounded as $x$ increases (i.e. stencils that are absolutely stable) for all $z < 0$. The three explicit stencils exhibit *conditional stability* in that for some $z$ the errors grow exponentially, while for other values the global errors remain bounded. For example, the forward Euler appears to be stable for $-2 < z < 0$ and unstable for $z < -2$.

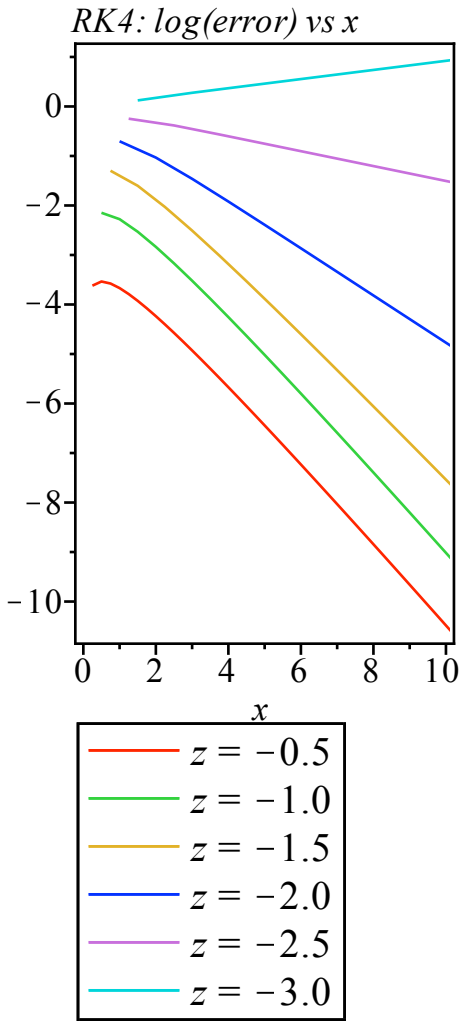
```
> for j from 1 to 5 do:
  q[j] := plot([seq(LogError(-2,-0.5*i,10,j),i=1..6)],x=0..10,axes=boxed,labels=[`x`,``],
  title=cat(Labels[j],`: log(error) vs x`),legend = [seq(typeset(z = -0.5*i),i=1..6)]);
  od:

  display(Array([[q[1],q[2],q[3]],[q[4],q[5],plot(x->NULL,axes=none)]]));
```


Forward Euler: log(error) vs x
2
0
−2
−4
−6
−8
0
2
4
6
8
10
x
z = −0.5
z = −1.0
z = −1.5
z = −2.0
z = −2.5
z = −3.0


Backward Euler: log(error) vs x
−1
−2
−3
−4
−5
−6
−7
0
2
4
6
8
10
x
z = −0.5
z = −1.0
z = −1.5
z = −2.0
z = −2.5
z = −3.0


Trapezoidal: log(error) vs x
−1
−2
−3
−4
−5
−6
−7
−8
−9
0
2
4
6
8
10
x
z = −0.5
z = −1.0
z = −1.5
z = −2.0
z = −2.5
z = −3.0



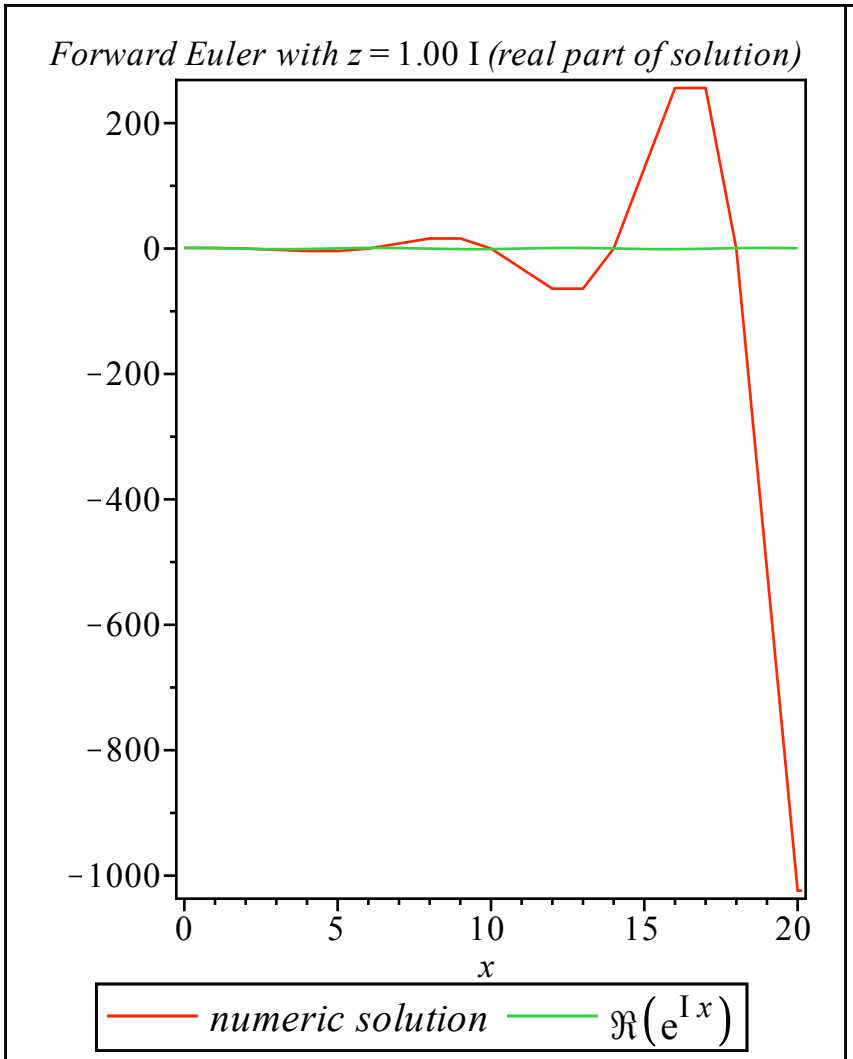

Now, the reader may be wondering what the practical import of this discussion is: while the explicit stencils have stability issues when $|z| = |\lambda| h$ is large and $\lambda < 0$, one can always obtain stable results when the stepsize is small enough. In other words, the explicit stencils will work as long as we are willing to wait long enough for results. However, the situation is a lot different if we consider imaginary $\lambda$. When $\lambda$ is complex, both the analytic solution $y(x) = e^{\lambda x}$ and the numeric solution will be complex, and hence difficult to plot. So in the plots below we will plot the real part of the numerical solutions, which ought to be an approximation to $\mathrm{Re}\left(e^{\lambda x}\right) = e^{\alpha x}\cos(\beta x)$ where

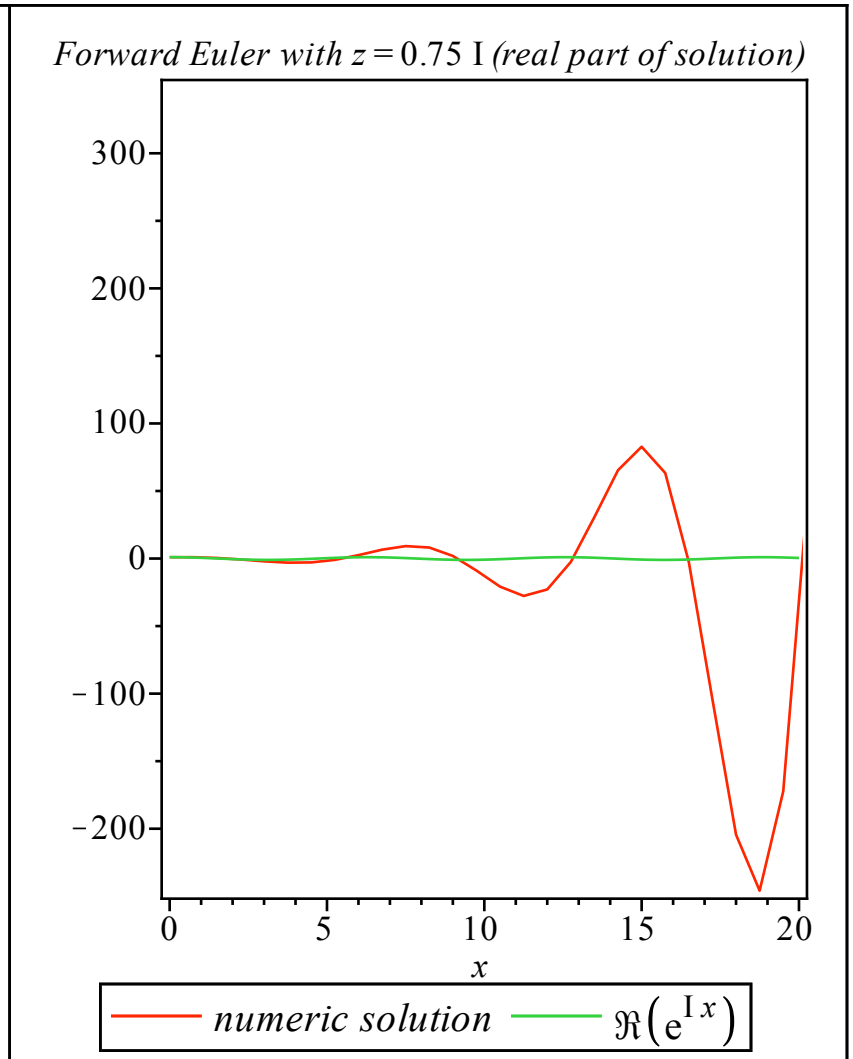
$\lambda = \alpha + i\beta$. The `plotter2` procedure accomplishes this:

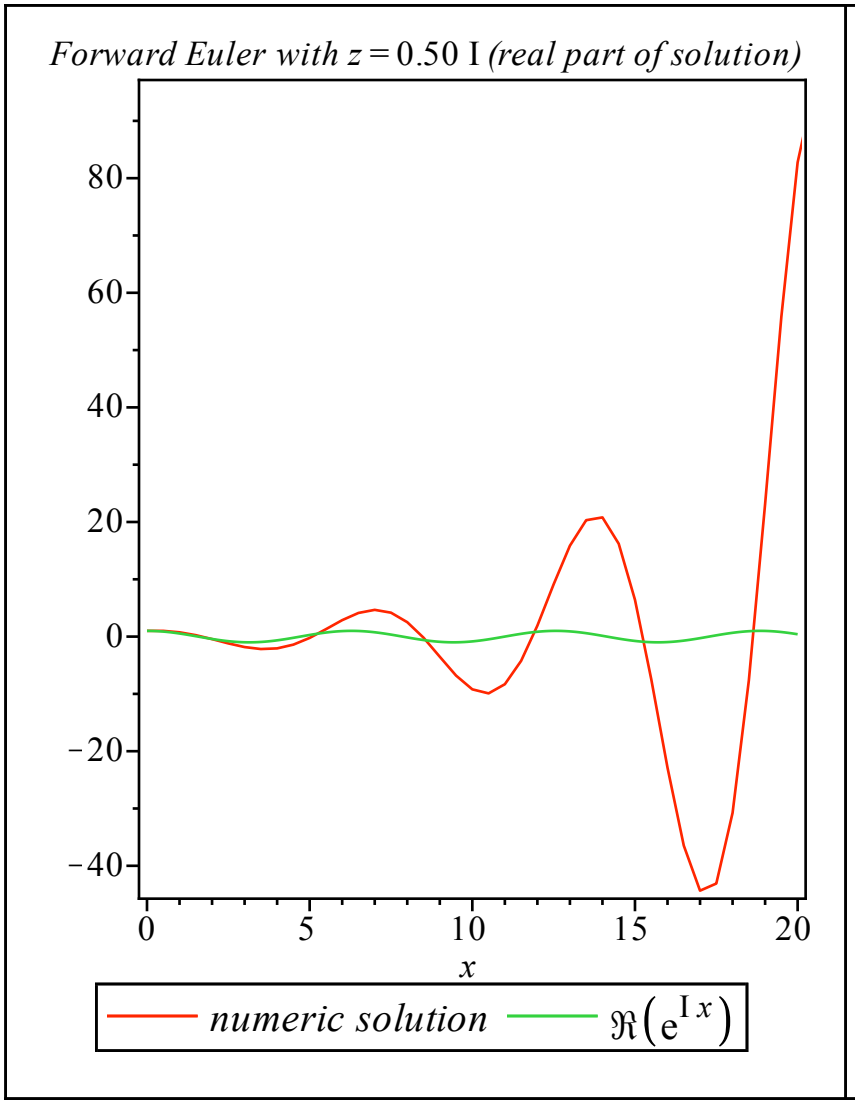
```
> plotter2 := (lambda,h,X,choice) -> plot([map(x->[x[1],Re(x[2])],Sol(lambda,h,X,choice)),Re
  (exp(lambda*x))],x=0..X,axes=boxed,title=typeset(Labels[choice],` with `,z = lambda*h,`
  (real part of solution)`),legend=[`numeric solution`,typeset(Re(exp(lambda*x)))]):
```

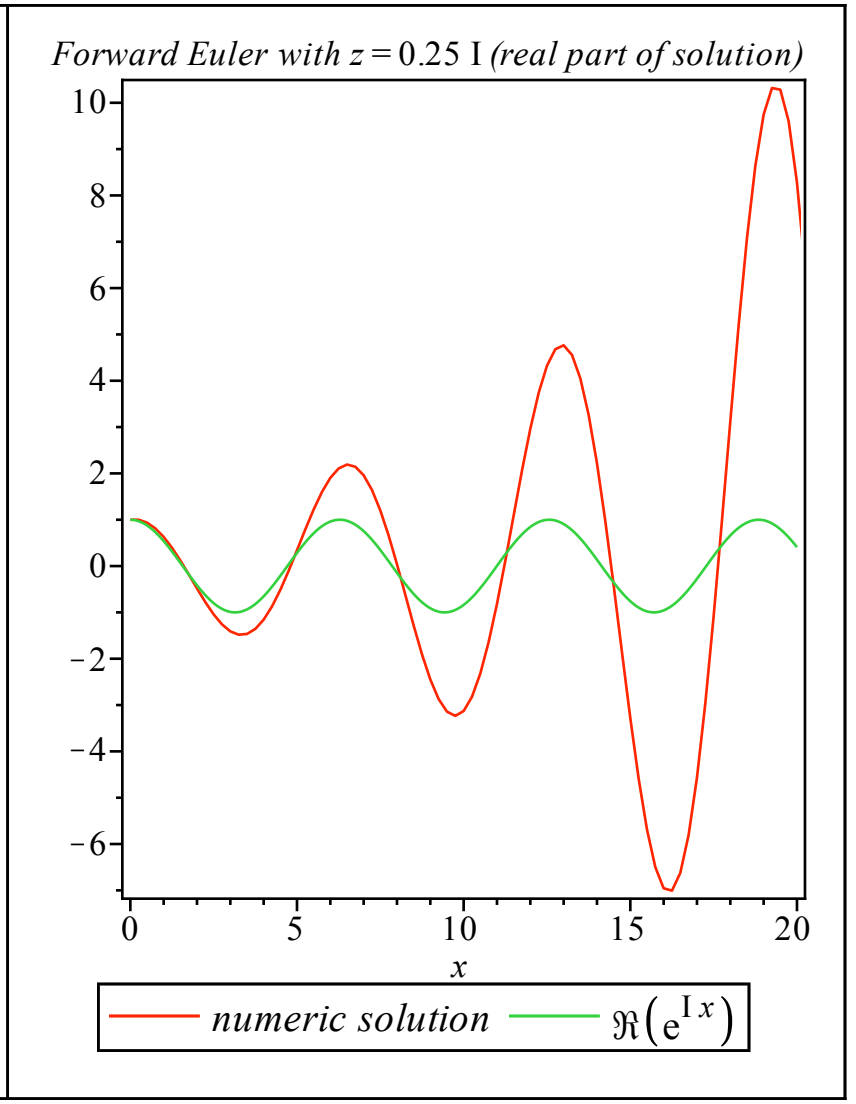
Here are some plots of the output of the forward Euler stencil when $\lambda$ is imaginary. In each plot, the numeric results appear to diverge from the expected analytic result for large $x$, irrespective of the choice of stepsize (or conversely, irrespective of the modulus of $z$). We say that the forward Euler stencil is unconditionally unstable when $\lambda$ is imaginary.

```
> p := 'p':
  for i from 1 to 4 do:
  p[i] := plotter2(I,1-0.25*(i-1),20,1);
  od:

  display(Matrix(2,2,convert(p,list)));
```


Forward Euler with z = 1.00 I (real part of solution)
200
0
-200
-400
-600
-800
-1000
0
5
10
15
20
x
numeric solution
ℜ(e^(I x))


Forward Euler with z = 0.75 I (real part of solution)
300
200
100
0
-100
-200
0
5
10
15
20
x
numeric solution
ℜ(e^(I x))

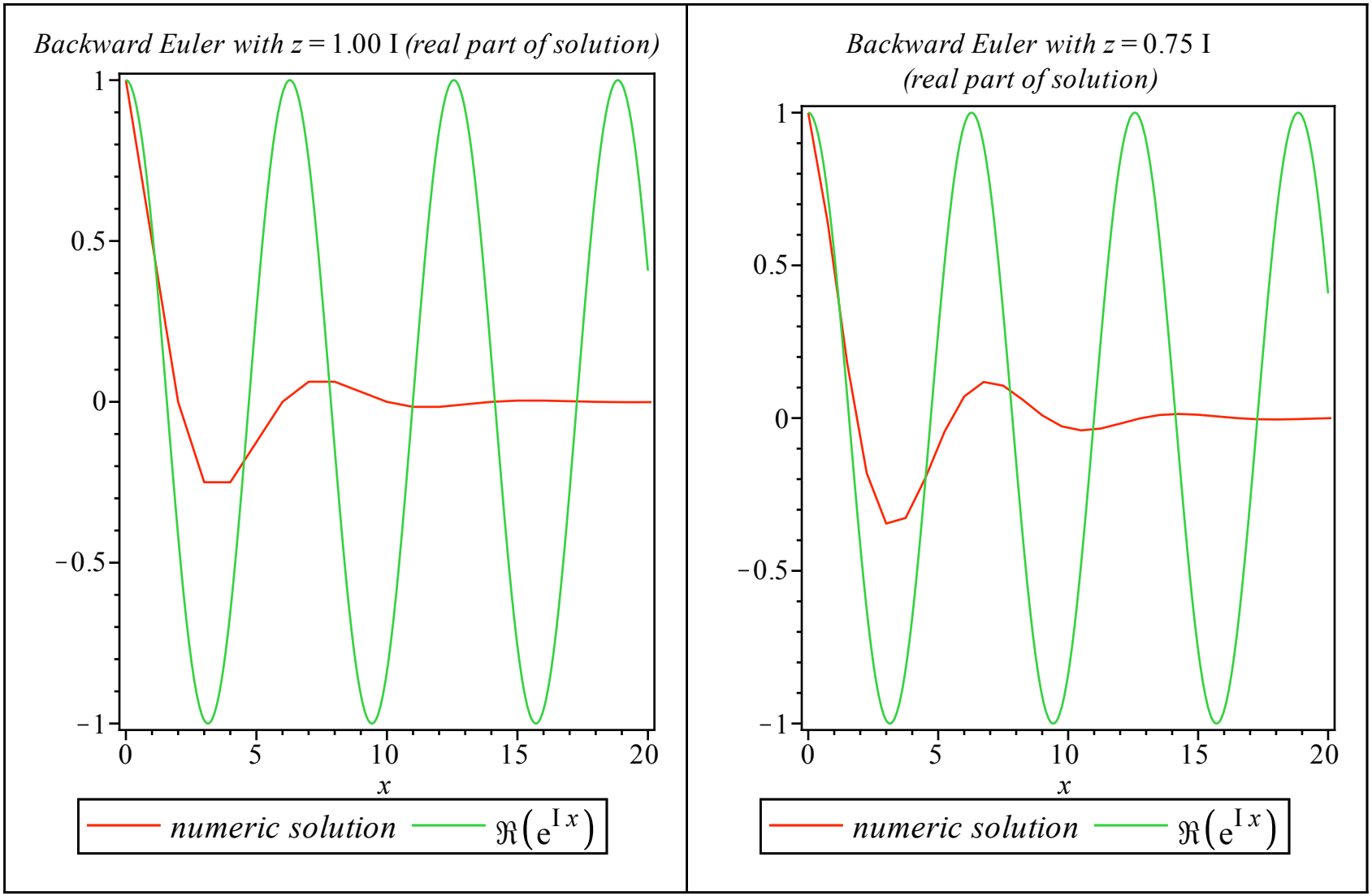


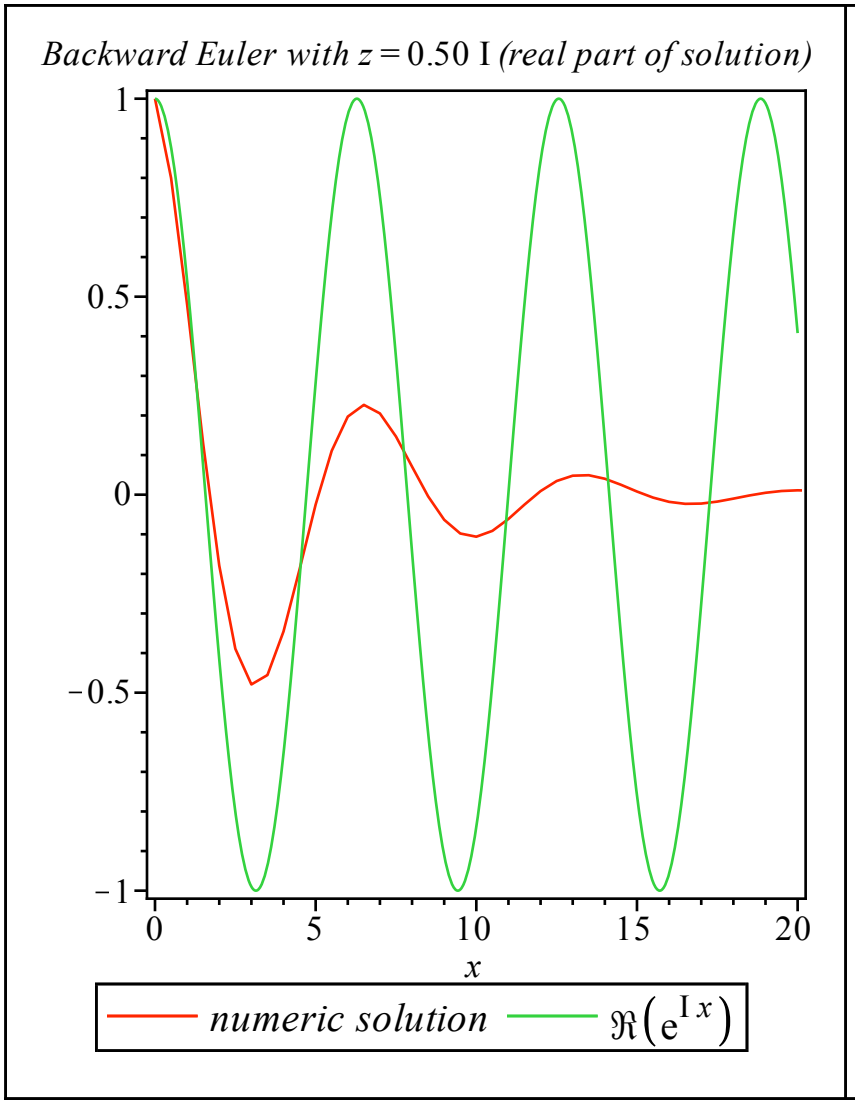



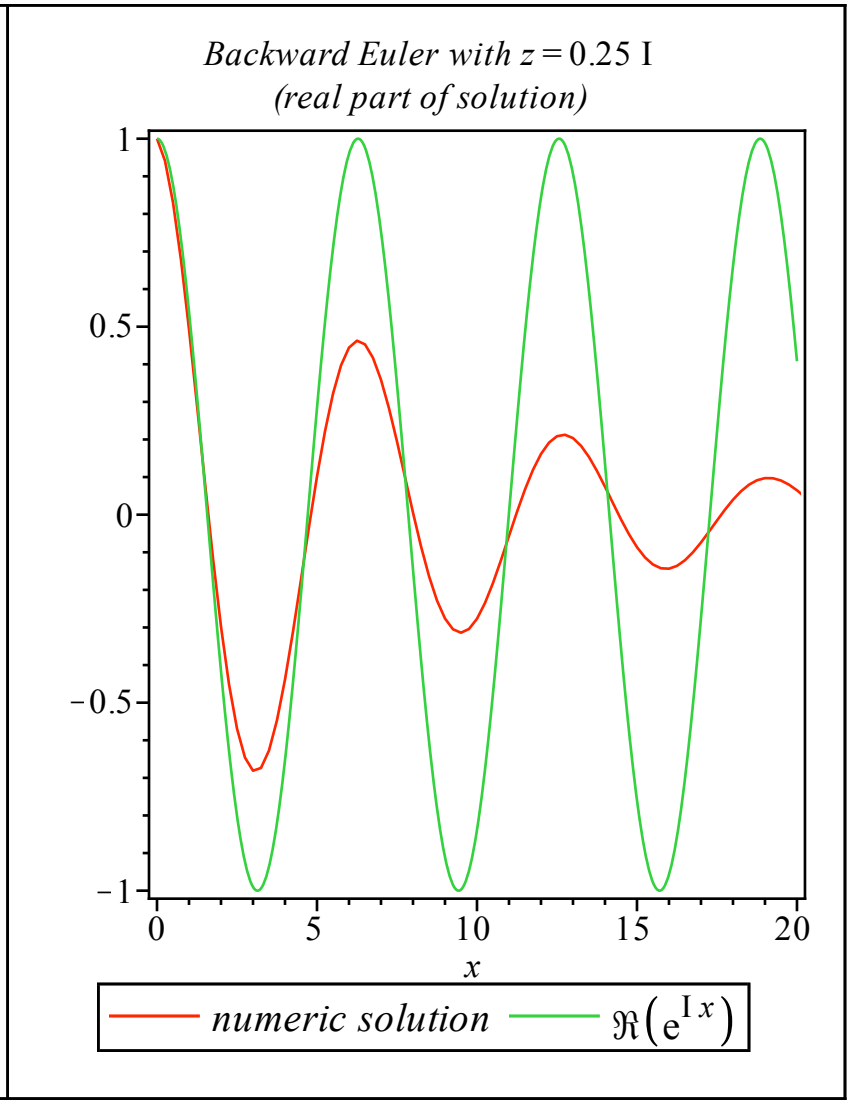

As for the case of $\lambda < 0$, the backward Euler stencil appears to be unconditionally stable for $\lambda$ imaginary (the numeric solution does not blow up). However, the stencil does not do a very good job of reproducing the analytic solution.

```
> p := 'p':
  for i from 1 to 4 do:
  p[i] := plotter2(I,1-0.25*(i-1),20,2);
  od:
```

```
display(Matrix(2,2,convert(p,list)));
```



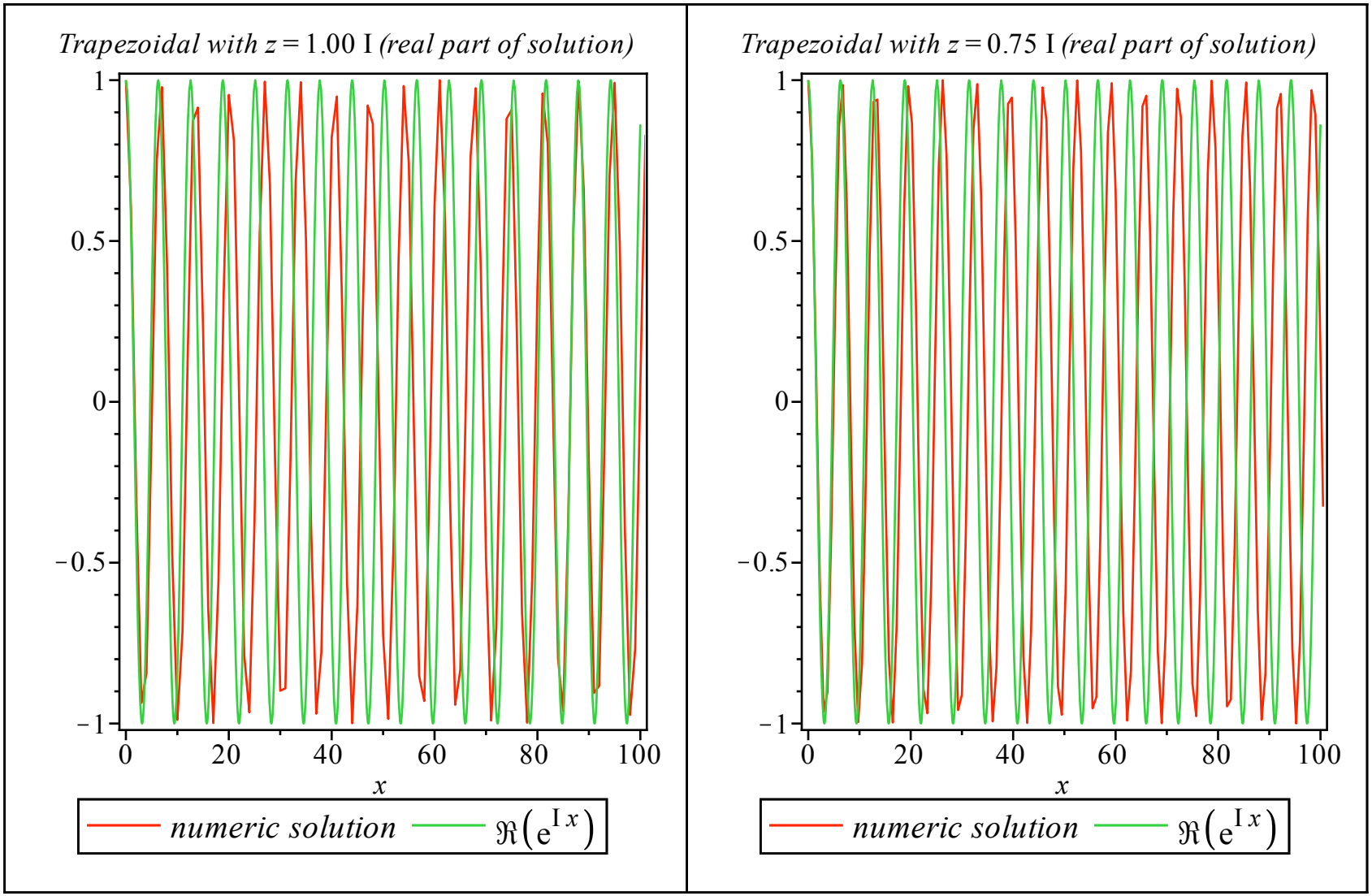


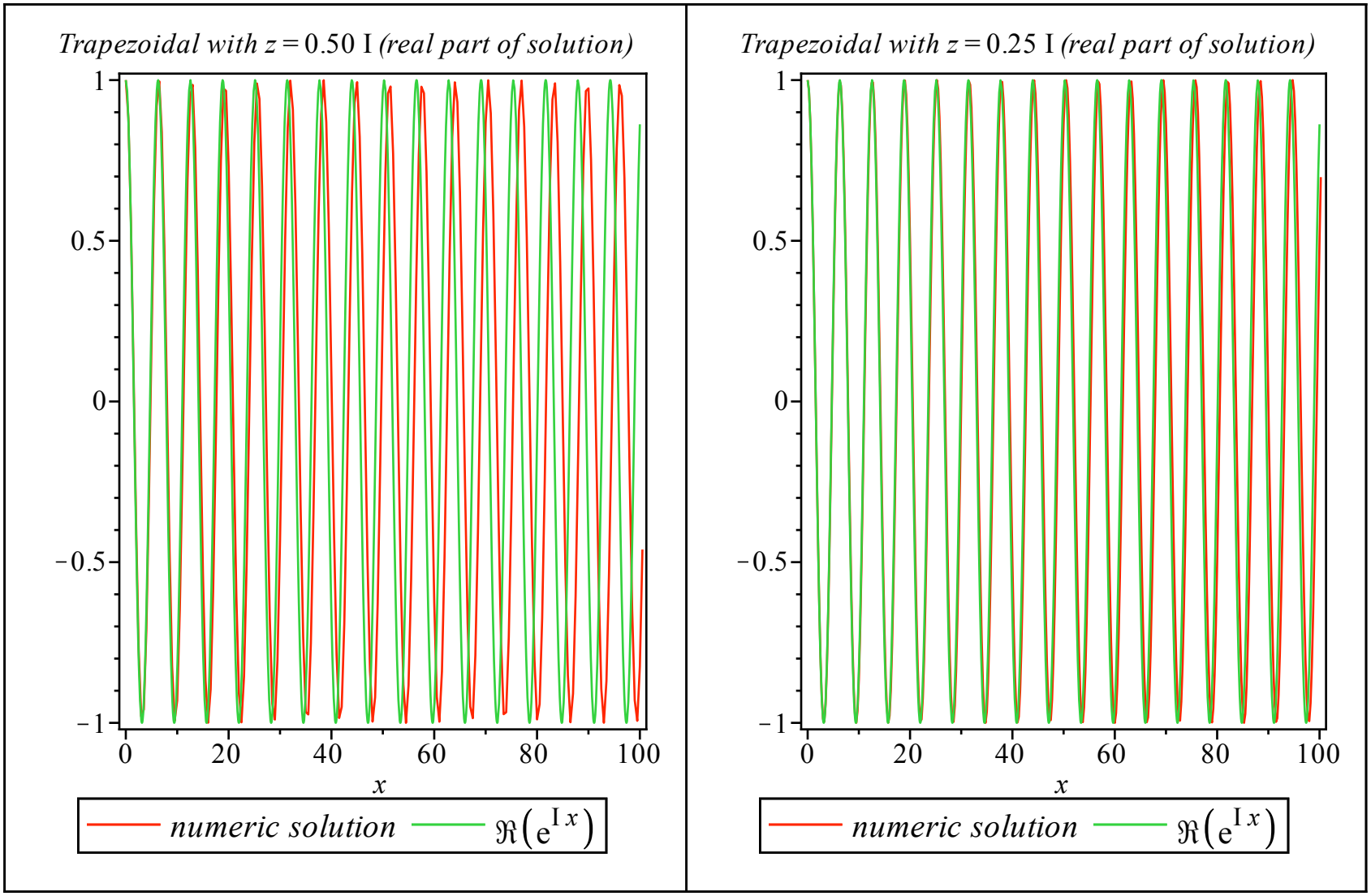




The trapezoidal method appears to be both stable and pretty good at reproducing the correct solution:

```
> p := 'p':
  for i from 1 to 4 do:
  p[i] := plotter2(I,1-0.25*(i-1),100,3);
  od:
```

```
display(Matrix(2,2,convert(p,list)));
```

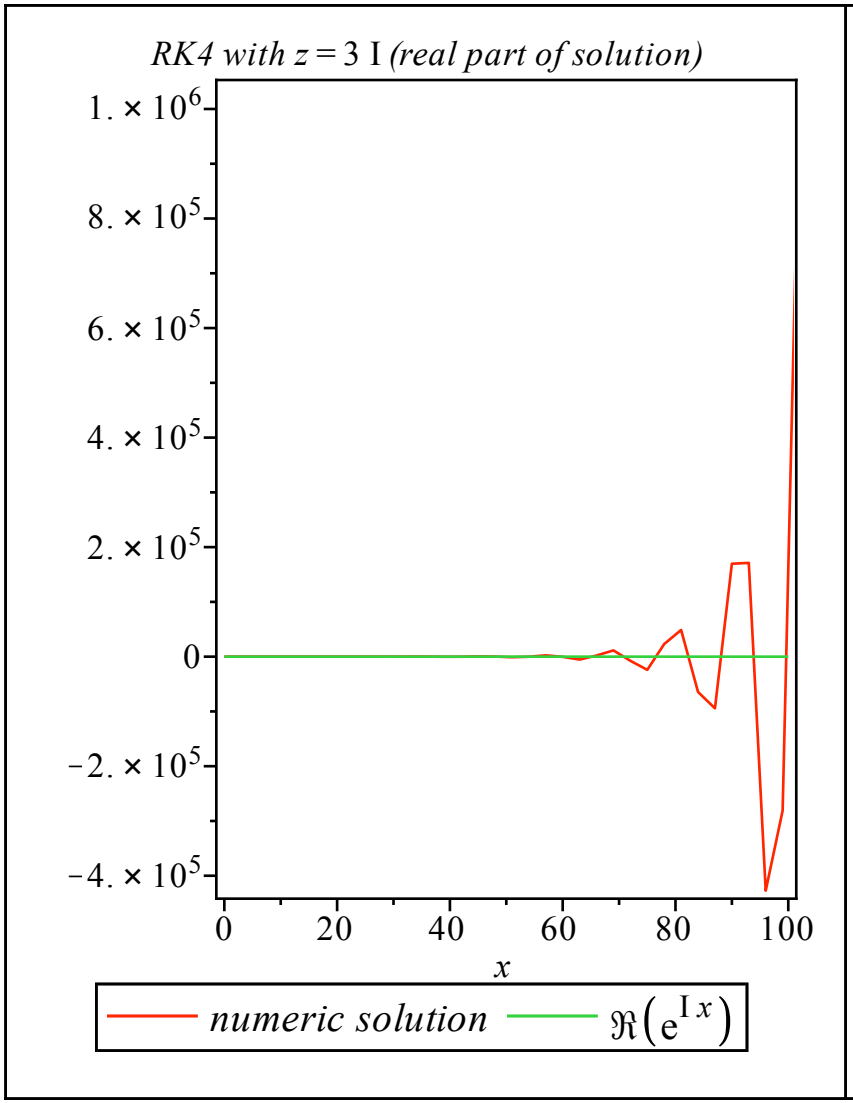


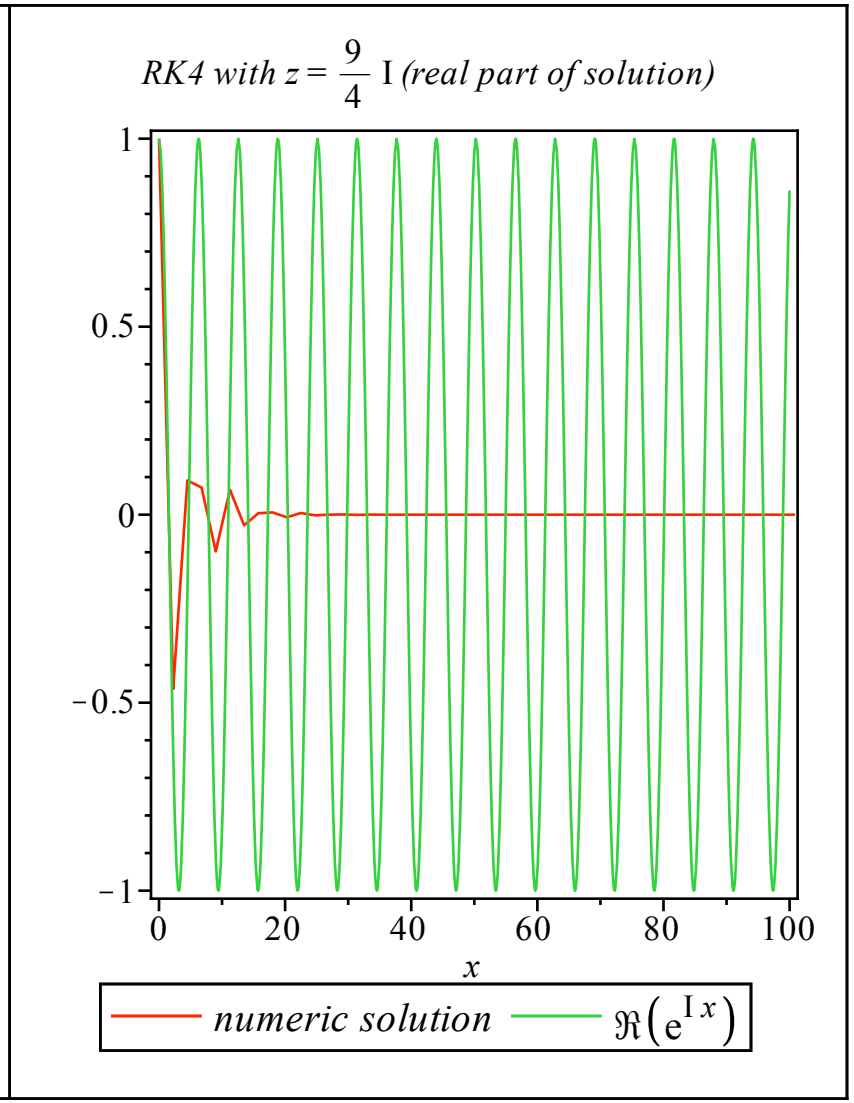


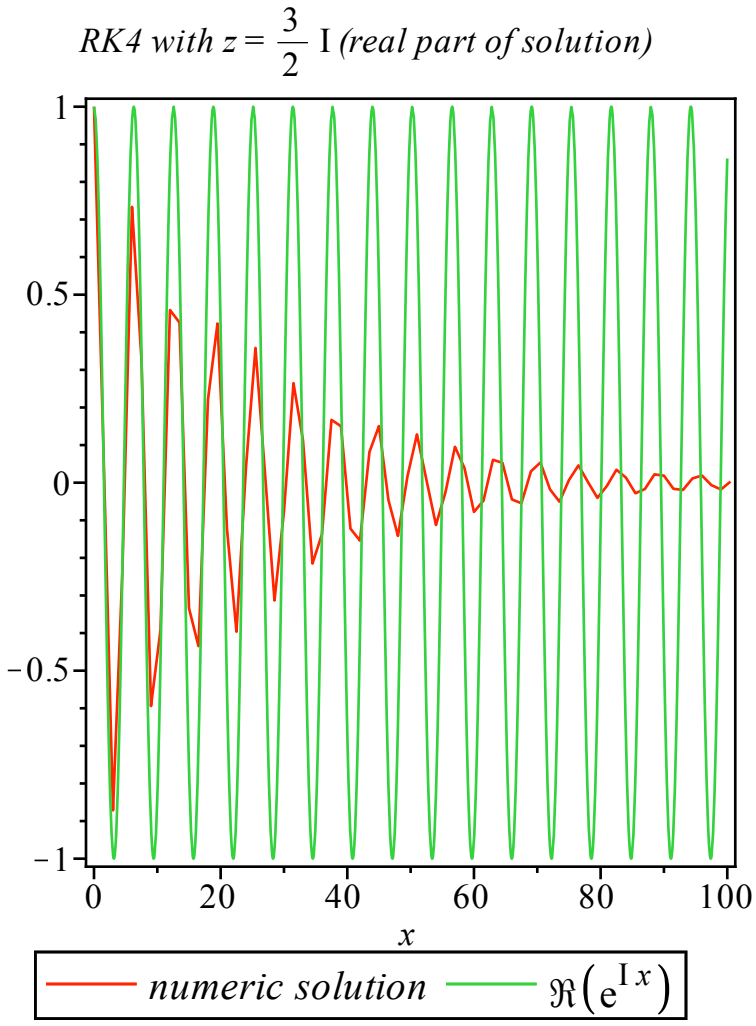

The fourth order Runge-Kutta stencil is interesting in that for some imaginary values of $z$ it appears unstable, while for others it looks stable That is, it appears to be *conditionally stable* for imaginary $z$.

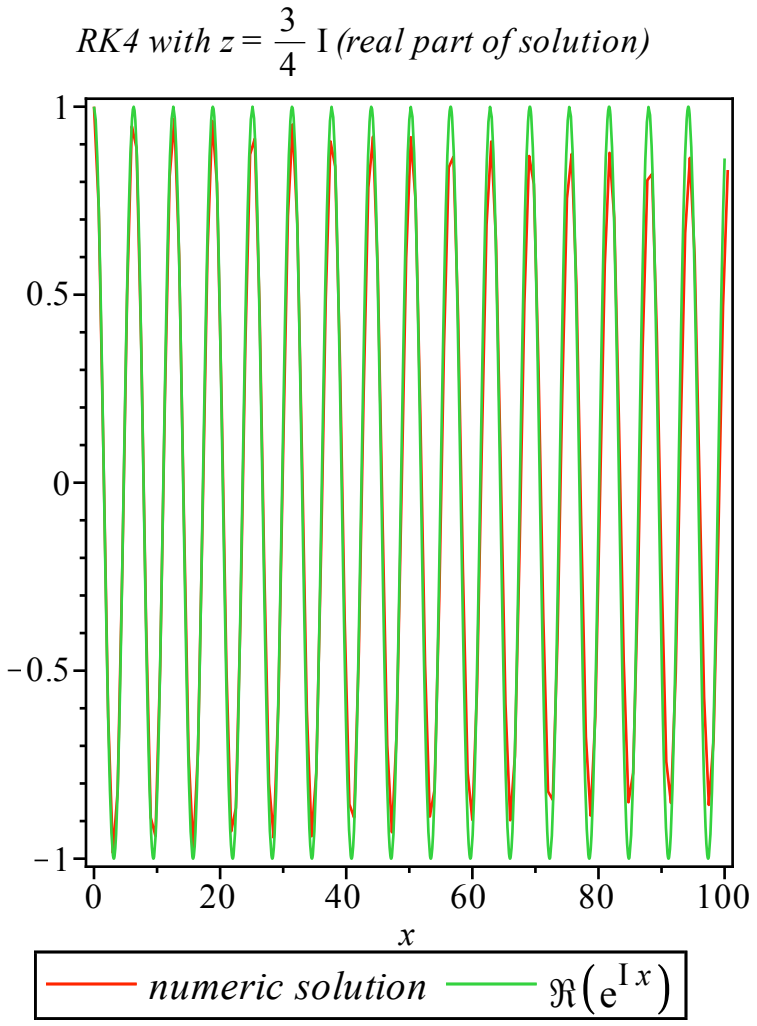
```
> p := 'p':
  for i from 1 to 4 do:
  p[i] := plotter2(I,3*(1-(i-1)/4),100,5);
  od:
```

```
display(Matrix(2,2,convert(p,list)));
```

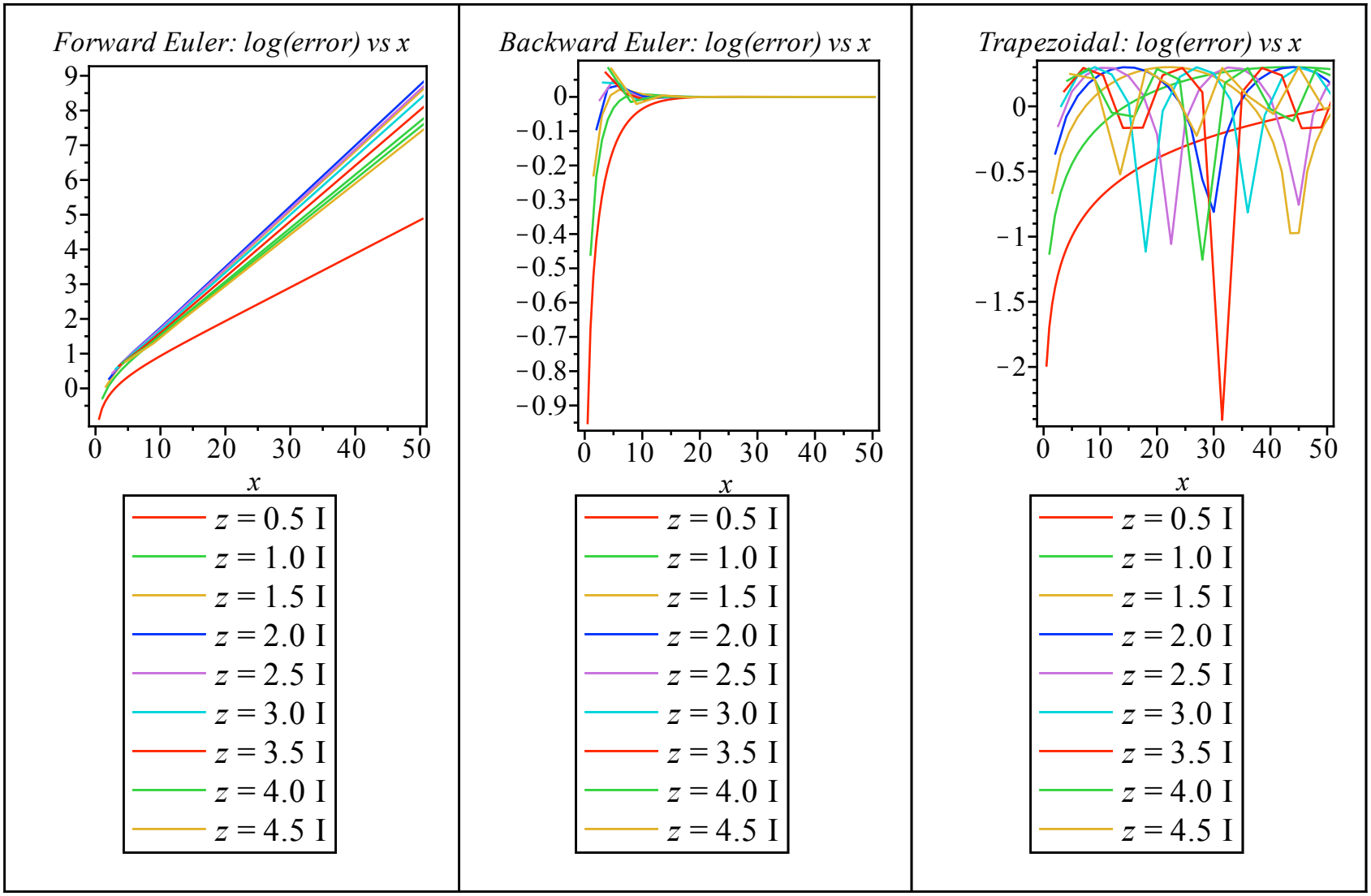


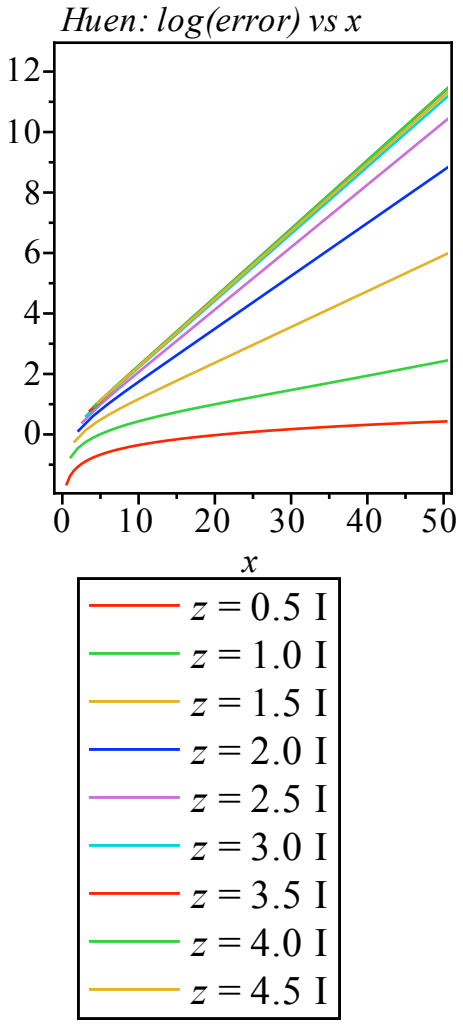



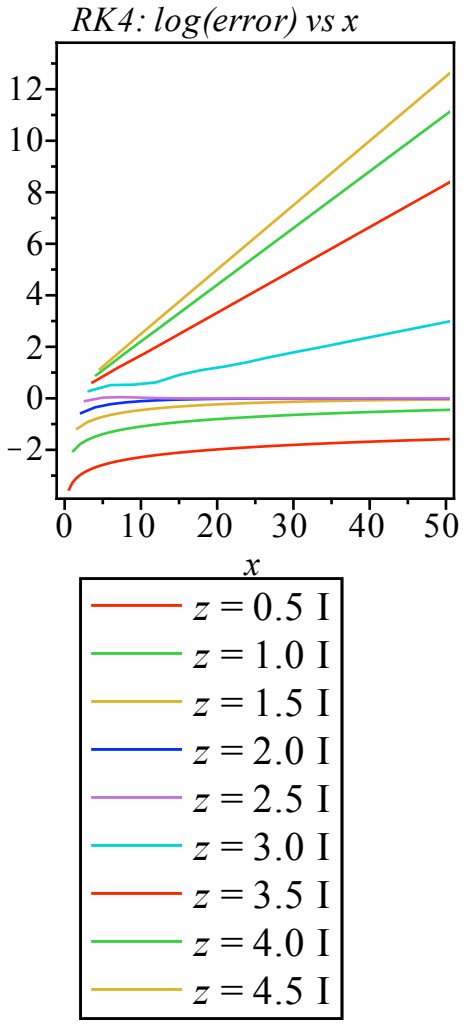



```
> for j from 1 to 5 do:
  q[j] := plot([seq(LogError(I,0.5*i*I,50,j),i=1..9)],x=0..50,axes=boxed,labels=[`x`,``],
  title=cat(Labels[j],`: log(error) vs x`),legend = [seq(typeset(z = 0.5*i*I),i=1..9)]);
  od:

  display(Array([[q[1],q[2],q[3]],[q[4],q[5],plot(x->NULL,axes=none)]]));
```


Forward Euler: log(error) vs x
9
8
7
6
5
4
3
2
1
0
0
10
20
30
40
50
x
z = 0.5 I
z = 1.0 I
z = 1.5 I
z = 2.0 I
z = 2.5 I
z = 3.0 I
z = 3.5 I
z = 4.0 I
z = 4.5 I
Backward Euler: log(error) vs x
0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
−0.8
−0.9
0
10
20
30
40
50
x
z = 0.5 I
z = 1.0 I
z = 1.5 I
z = 2.0 I
z = 2.5 I
z = 3.0 I
z = 3.5 I
z = 4.0 I
z = 4.5 I
Trapezoidal: log(error) vs x
0
−0.5
−1
−1.5
−2
0
10
20
30
40
50
x
z = 0.5 I
z = 1.0 I
z = 1.5 I
z = 2.0 I
z = 2.5 I
z = 3.0 I
z = 3.5 I
z = 4.0 I
z = 4.5 I




## Stability regions for the test problem $y' = \lambda y$

```
> lambda := 'lambda':
  f := 'f':
  i := 'i':
```

```
P := 'P':
q := 'q':
```

We would now like to attempt to explain some of the behaviour witnessed in the previous section. We will need to have expressions for each of the above stencils that include the one-step error, which we call **_Stencil**. These are generated by the following code:

```
> derivative[1] := D(y)(x) = f(x,y(x)):
  for j from 2 to 5 do:
       derivative[j] := subs(derivative[1],convert(diff(derivative[j-1],x),D)):
  od:

  Subs := [y_new = y(x+h),y_old=y(x),x_new=x+h,x_old=x]:

  for j from 1 to 5 do:
     one_step[j] := O(h^ldegree(convert(simplify(subs(convert(derivative,list),series(subs
  (Subs,lhs(Stencil[j])),h))),polynom),h)):
     _Stencil[j] := subs(Subs,lhs(Stencil[j]))=one_step[j];
     print(Labels[j],_Stencil[j]);
  od:
```

$$\textit{Forward Euler}, y(x+h) - y(x) - f(x, y(x))\, h = \mathrm{O}\left(h^2\right)$$

$$\textit{Backward Euler}, y(x+h) - y(x) - f(x+h, y(x+h))\, h = \mathrm{O}\left(h^2\right)$$

$$\textit{Trapezoidal}, y(x+h) - y(x) - \frac{1}{2} f(x, y(x))\, h - \frac{1}{2} f(x+h, y(x+h))\, h = \mathrm{O}\left(h^3\right)$$

$$\textit{Huen}, y(x+h) - y(x) - h\left(\frac{1}{2} f(x, y(x)) + \frac{1}{2} f(x+h, y(x) + f(x, y(x))\, h)\right) = \mathrm{O}\left(h^3\right)$$

$$\textit{RK4}, y(x+h) - y(x) - h\left(\frac{1}{6} f(x, y(x)) + \frac{1}{3} f\left(x + \frac{1}{2} h, y(x) + \frac{1}{2} f(x, y(x))\, h\right) + \frac{1}{3} f\left(x + \frac{1}{2} h, y(x)\right.\right. \quad \textbf{(2.1}$$

$$\left. + \frac{1}{2} h f\left(x + \frac{1}{2} h, y(x) + \frac{1}{2} f(x, y(x))\, h\right)\right) + \frac{1}{6} f\left(x + h, y(x) + h f\left(x + \frac{1}{2} h, y(x) + \frac{1}{2} h f\left(x + \frac{1}{2} h, y(x)\right.\right.\right.$$

$$\left.\left.\left. + \frac{1}{2} f(x, y(x))\, h\right)\right)\right)\right) = \mathrm{O}\left(h^5\right)$$

Let us to return to our test problem $y'(x) = \lambda y(x)$; i.e. $f(x, y) = \lambda y$. Now, we will denote the error in our numerical solution at the $i^{th}$ to b $E_i = y_i - y(x_i)$. If we subtract each element in **_Stencil** **(2.1)** from each element in **Stencil** **(1.1)** and make use of the definition of $E_i$, we obtain:

```
> f := (x,y) -> lambda*y;
  E_def := E[i] = y[i] - y(x[i]);
  Subs := [y(x+h)=y(x[i+1]),y(x)=y(x[i]),y_new=y[i+1],y_old=y[i],isolate(E_def,y[i]),subs(i=
  i+1,isolate(E_def,y[i]))];
  for j from 1 to 5 do:
     EOM[j] := collect(simplify(subs(Subs$2,Stencil[j]-_Stencil[j])),[E[i+1],E[i]]);
  od;
```

$$f := (x, y) \to \lambda\, y$$

$$E\_def := E_i = y_i - y(x_i)$$

$$Subs := \left[y(x+h) = y(x_{i+1}), y(x) = y(x_i), y\_new = y_{i+1}, y\_old = y_i, y_i = E_i + y(x_i), y_{i+1} = E_{i+1} + y(x_{i+1})\right]$$

$$EOM_1 := E_{i+1} + (-1 - \lambda\, h)\, E_i = -O(h^2)$$

$$EOM_2 := (1 - \lambda\, h)\, E_{i+1} - E_i = -O(h^2)$$

$$EOM_3 := \left(1 - \frac{1}{2}\lambda\, h\right) E_{i+1} + \left(-1 - \frac{1}{2}\lambda\, h\right) E_i = -O(h^3)$$

$$EOM_4 := E_{i+1} + \left(-1 - \lambda\, h - \frac{1}{2}\lambda^2 h^2\right) E_i = -O(h^3)$$

$$EOM_5 := E_{i+1} + \left(-\frac{1}{6}\lambda^3 h^3 - 1 - \lambda\, h - \frac{1}{2}\lambda^2 h^2 - \frac{1}{24}\lambda^4 h^4\right) E_i = -O(h^5) \qquad \textbf{(2.2}$$

`EOM[j]` tells us how the error at the $(i+1)^{th}$ node is related to the error at the $i^{th}$ node for the $j^{th}$ stencil. Each of these can be re-written as

$$E_{i+1} = P(z)\, E_i + O(h^p), \quad z = h\lambda.$$

For the explicit stencils, $P(z)$ is a polynomial; for the implicit stencils it is a rational function:

```
> for j from 1 to 5 do:
      P_def[j] := P(z) = subs(lambda=z/h,solve(lhs(EOM[j]),E[i+1])/E[i]);
  od;
```

$$P\_def_1 := P(z) = 1 + z$$

$$P\_def_2 := P(z) = -\frac{1}{-1 + z}$$

$$P\_def_3 := P(z) = -\frac{2+z}{-2+z}$$

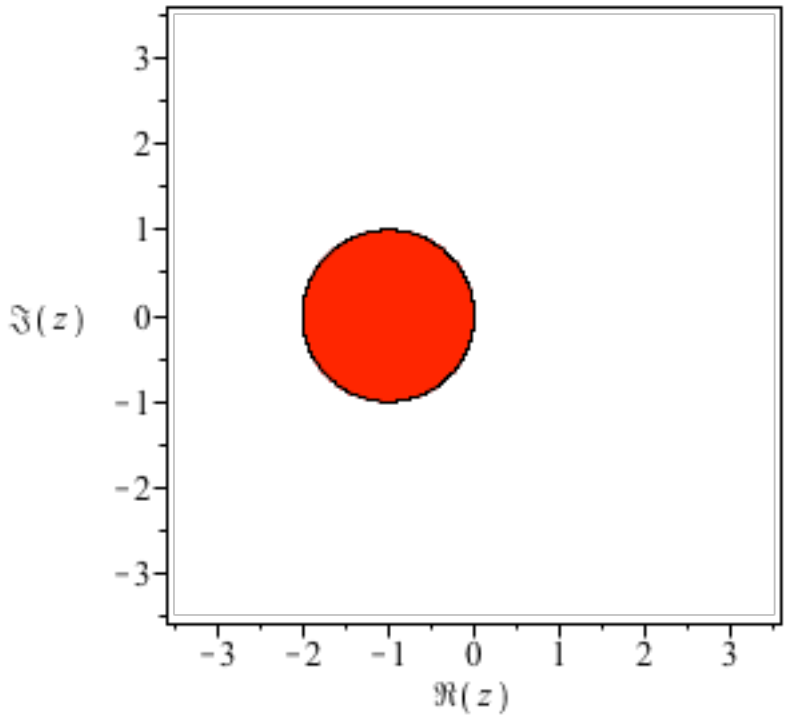
$$P\_def_4 := P(z) = 1 + z + \frac{1}{2}z^2$$

$$P\_def_5 := P(z) = \frac{1}{6}z^3 + 1 + z + \frac{1}{2}z^2 + \frac{1}{24}z^4 \qquad \textbf{(2.3}$$

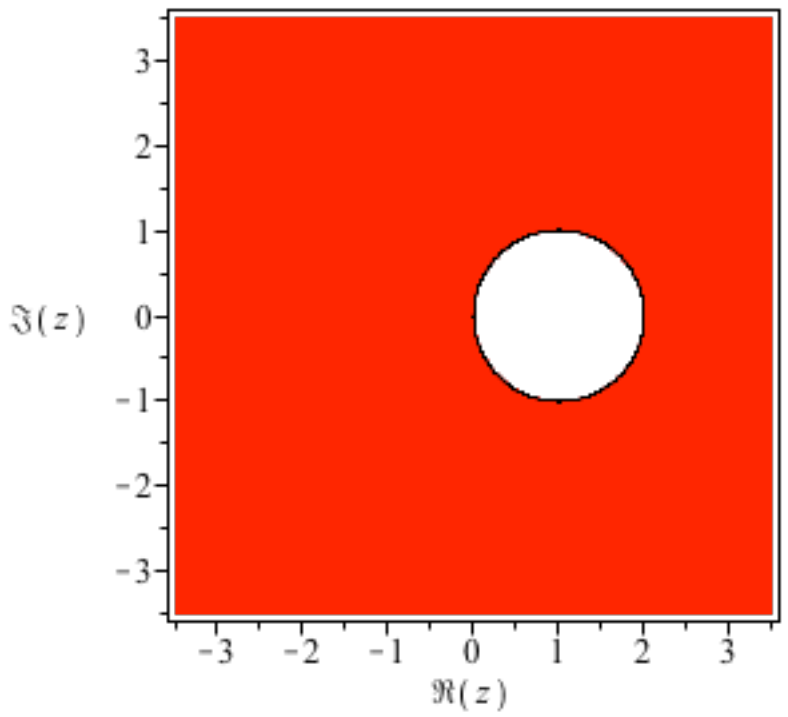
We see that if $|P(z)| > 1$, the errors will grow exponentially with $i$ (or $x$). Hence we say a stencil is absolutely stable if $|P(z)| \leq 1$. (Different authors sometimes employ slightly different definitions; for example, one could define absolute stability by $|P(z)| < 1$.) Note that even if a stencil is absolutely stable, it does not mean that the errors do not grow as the simulation progresses; the $O(h^p)$ terms will have some bearing on the evolution of the errors. However, in practice the condition $|P(z)| \leq 1$ *usually* implies that the errors do not blow-up expontially as the simulation progresses provided that the true solution itself does not diverge. Now we plot the regions in the complex $z$ plane for which each stencil is stable (i.e. for which $|P(z)| \leq 1$):

```
> for j from 1 to 5 do:
  _P[j] := unapply(rhs(P_def[j]),z);
  od:

  p := 'p':
  for j from 1 to 5 do:
  p[j] := implicitplot(abs(_P[j](x+I*y))-1,x=-3.5..3.5,y=-3.5..3.5,filled=true,grid=[5,5],
  gridrefine=6,axes=boxed,scaling=constrained,title=cat(`Stability region for the `,Labels
  [j],` stencil`),labels=[Re(z),Im(z)],coloring=[red,white]);
  od:
  p[6] := plot(x->NULL,axes=none):
  display(Matrix(3,2,convert(p,list)));
```

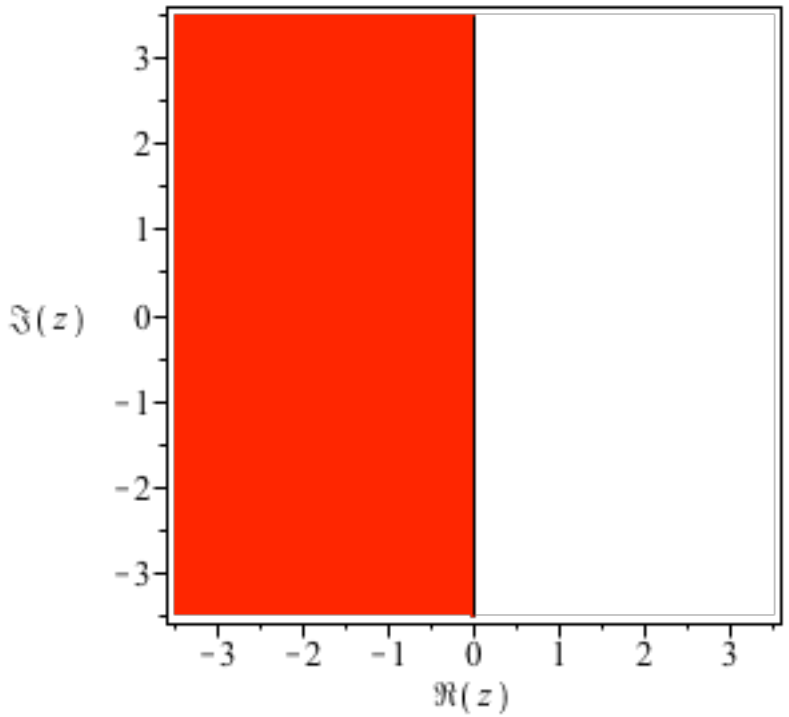
*Stability region for the Forward Euler stencil*



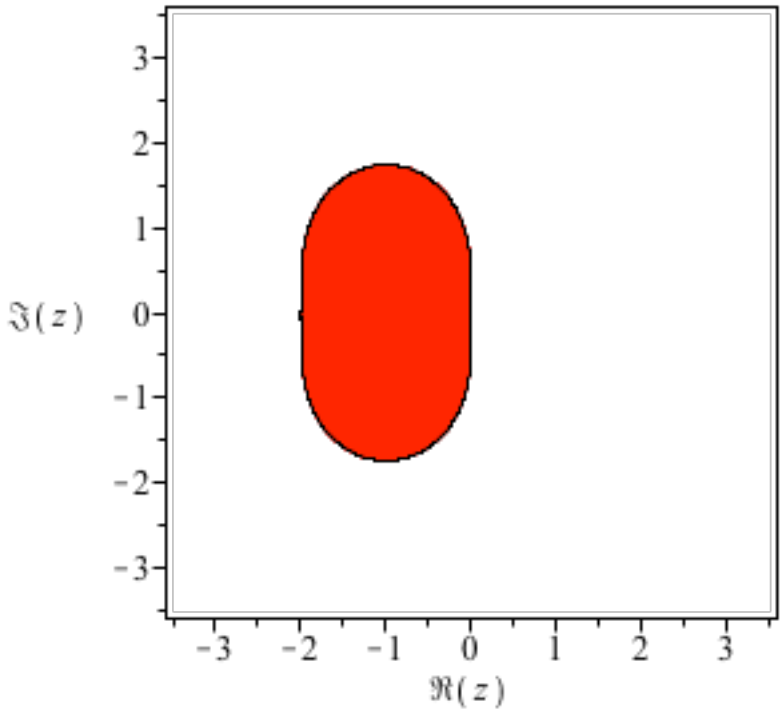

*Stability region for the Backward Euler stencil*

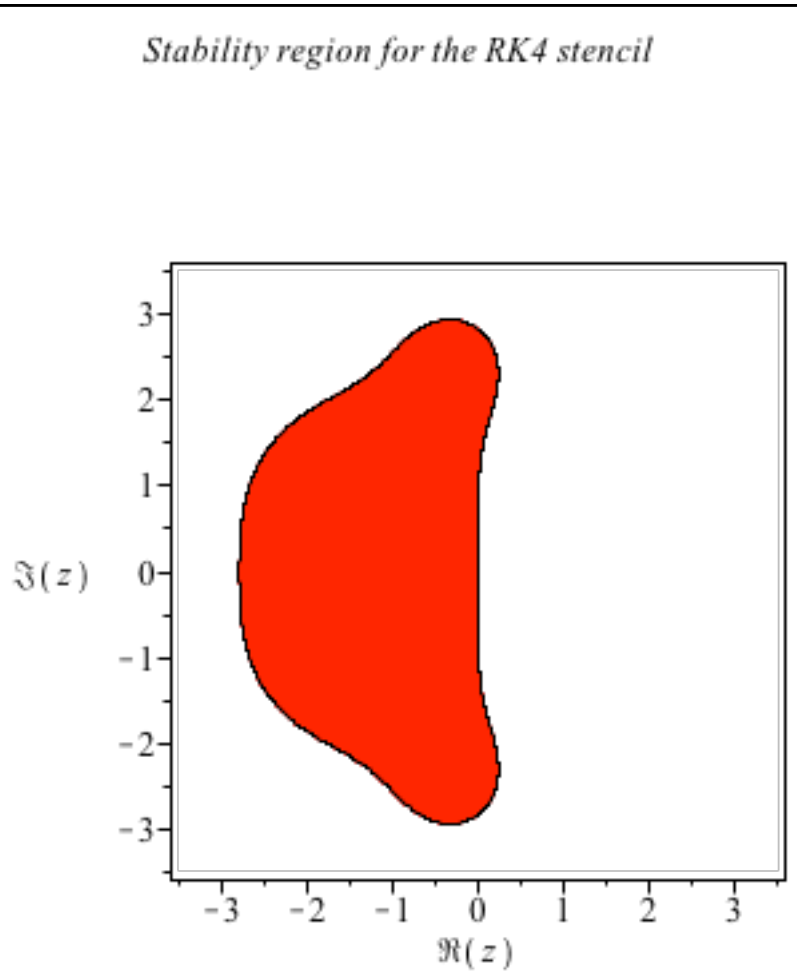


*Stability region for the Trapezoidal stencil*



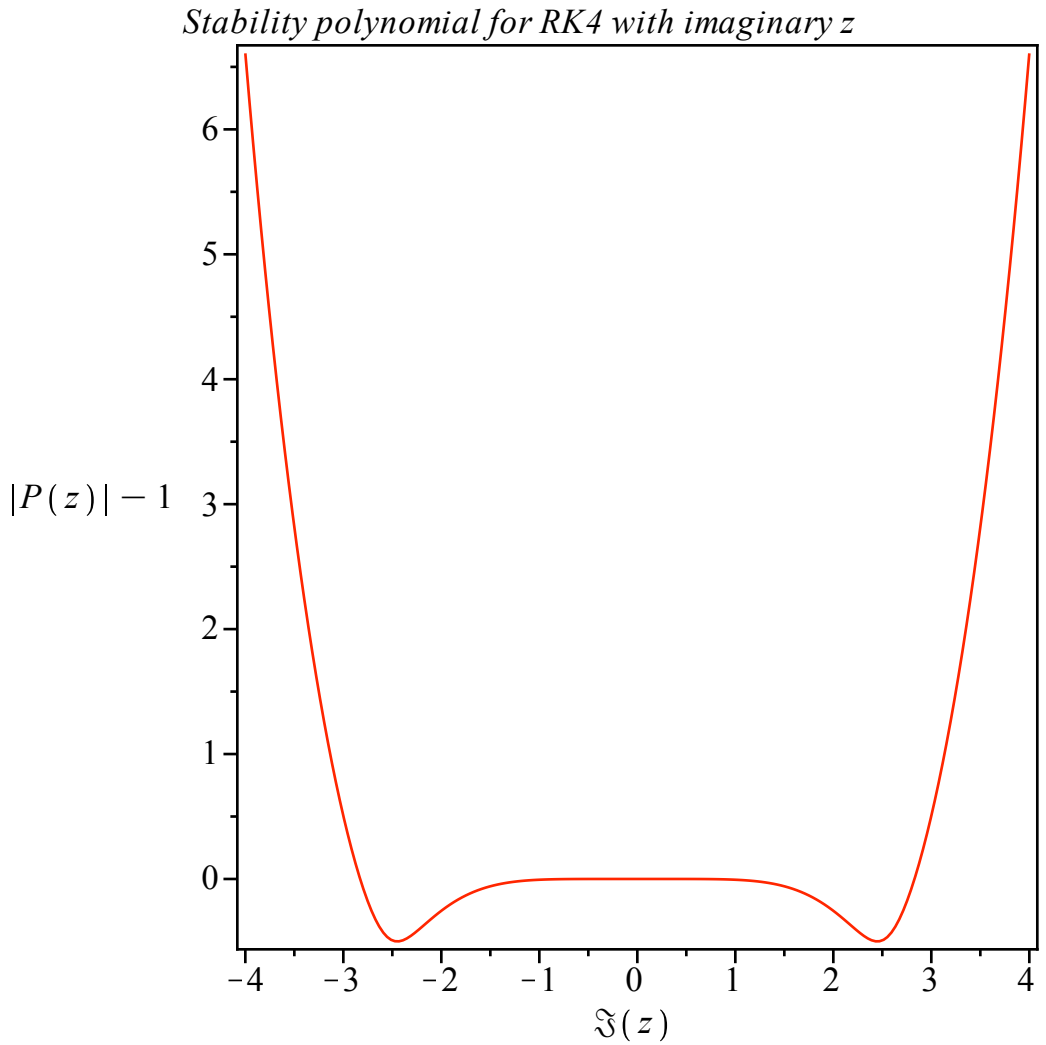

*Stability region for the Huen stencil*






In these plots, the red region shows where each stencil is stable. The shape of the stability regions explains many of the features seen in the previous section: For example, it is clear that the forward Euler stencil is stable on the real axis for $z \in [-2, 0]$ and unstable for all finite $z$ on the imaginary axis. It is interesting to actually plot $|P(z)| - 1$ for imaginary $z$ for the RK4 stencil:

```
> plot(abs(_P[5](I*y))-1,y=-4..4,axes=boxed,labels=[Im(z),abs(P(z))-1],title=`Stability
  polynomial for RK4 with imaginary z`);
```



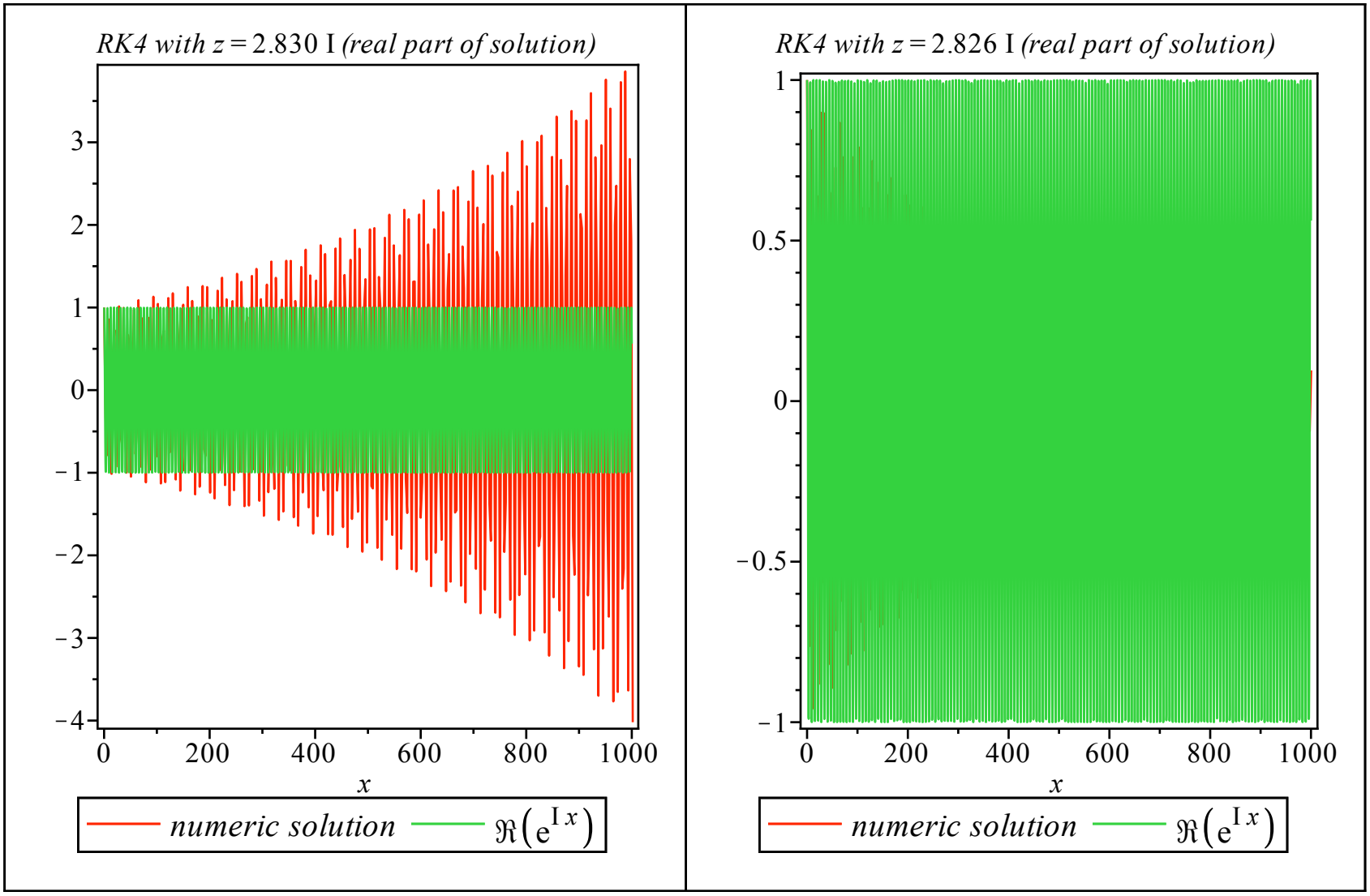

We can see visually that the RK4 stencil will be stable for $z = iq$, with $-2.75 \lesssim q \lesssim +2.75$. We can determine a much more accurate stability threshold using **fsolve**:

```
> fsolve(abs(_P[5](I*x))-1,x=2.75);
```

2.828427125 **(2.4**

Here are two simulations with $z$ close to this stability threshold. On the left is a marginally unstable simulation, on the right is a marginally stable one.

```
> display(Matrix([plotter2(I,2.830,1000,5),plotter2(I,2.826,1000,5)]));
```




Finally, we can comment on the rather uncanny ability for the trapezoidal method to obtain much better results than the other stencils when $z$ is imaginary. Note that all of the stencils can be written as

$$y_{i+1} = P(z)\, y_i.$$

For the trapezoidal stencil with $z = iq$ this is explicitly:

```
> Trap := y[i+1] = _P[3](I*q)*y[i];
```

$$Trap := y_{i+1} = -\frac{(2 + I\,q)\,y_i}{-2 + I\,q} \tag{2.5}$$

Taking absolute values of each side reveals

```
> simplify(map(x->abs(x),Trap)) assuming(q,real);
```

$$|y_{i+1}| = |y_i| \tag{2.6}$$

That is, the absolute value of the numerical solution is conserved as the simulation procedes. The replicates a very important property of the exact solution $y(x) = e^{iqx}$ namely $|y(x)| = 1$.

# Stability for nonlinear problems

We conclude by looking at the case of general choices of $f(x, y)$; that is, we allow for the possibiity that $y' = f(x, y)$ is a nonlinear ODE foı $y(x)$. It turns out that the conclusions of the stability analysis are applicable if we restrict ourselves to regions where the true solution is slowly varying. We first demonstrate how this works somewhat heuristically. For general $f(x, y)$, we are trying to solve the following ODE:

```
> f := 'f':
  ode := diff(y(x),x) - f(x,y(x));
```

$$ode := \frac{d}{dx}\,y(x) - f(x, y(x)) \tag{3.1}$$

Let's assume that for $x$ in some interval, the true solution is approximately constant $y(x) \approx y_0$. We can then represent $y(x)$ as a perturbative series:

```
> perturbation := y(x) = y[0] + add(epsilon^(j)*v[j](x),j=1..3);
```

$$perturbation := y(x) = y_0 + \varepsilon\,v_1(x) + \varepsilon^2\,v_2(x) + \varepsilon^3\,v_3(x) \tag{3.2}$$

Here, $\varepsilon \ll 1$ is a perturbative expansion parameter. Plugging this into the ODE and retaining the first couple of terms in the series yields aı ODE for the first order perturbation:

```
> first_order := convert(series(subs(perturbation,ode),epsilon,2),polynom);
  first_order := isolate(first_order,D(v[1])(x));
```

$$first\_order := -f(x, y_0) + \left(D(v_1)(x) - D_2(f)\,(x, y_0)\,v_1(x)\right)\varepsilon$$

$$first\_order := D(v_1)(x) = \frac{f(x, y_0)}{\varepsilon} + D_2(f)\,(x, y_0)\,v_1(x) \tag{3.3}$$

We now make the further approximation that both $f$ and $\frac{\partial}{\partial y} f$ are constant in the interval of interest:

```
> approximations := [f(x,y[0])=mu*epsilon,D[2](f)(x,y[0])=lambda];
```

$$approximations := \left[ f\left(x, y_0\right) = \mu\, \varepsilon,\ D_2(f)\left(x, y_0\right) = \lambda \right] \quad \textbf{(3.4}$$

Putting these into the equation of motion for $v_1(x)$ yields:

```
> first_order := subs(approximations,first_order);
```

$$first\_order := \mathrm{D}\left(v_1\right)(x) = \mu + \lambda\, v_1(x) \quad \textbf{(3.5}$$

Finally, let's change variables to $V_1(x) = v_1(x) + \frac{\mu}{\lambda}$:

```
> first_order := isolate(expand(subs(v[1](x)=V[1](x)-mu/lambda,convert(first_order,diff))),
  diff(V[1](x),x));
```

$$first\_order := \frac{\mathrm{d}}{\mathrm{d}x} V_1(x) = \lambda\, V_1(x) \quad \textbf{(3.6}$$

The claim is now that solving **(3.1)** numerically over a region where $f$ and $\frac{\partial}{\partial y} f$ are slowly varying is approximately equivalent to solving **(3.6)**; i.e., the numeric stencil is really solving a perturbative problem to lowest order. But the latter is entirely equivalent to the test problem which means that the stability results of the previous section should carry over to the nonlinear case provided that our approximations are valid. It is easy enough to see that if the step size is sufficiently small compared to the scales over which $y(x)$ and $f(x, y(x))$ vary, the "slowly varying" condition will be satisfied. Now, it may not be immediately clear to the reader that the fact that solving **(3.6)** is indeed eqivalent to solving **(3.1)** from a numerical point of view: It is plausible that if the true solution is not much different than a constant, the numerical algorithm is indeed finding the first order perturbation, but is that what is really going on? To answer this, we repeat some of the calculations of the previous section without the assumption $y' = \lambda y$. As before, we will define the error $E_i$ as the difference between numeric and actual solutions at the $i^{th}$ node:

```
> E_def := epsilon * E[i] = y[i] - y(x[i]);
```

This is actually a little different from what we had before [*c.f.* **(2.2)**] because of the ε factor: this is included for book-keeping purposes and we can later set it equal to unity. We now derive the equations of motions for the errors as before by subtracting the elements of `_Stencil` from `Stencil`.

```
> Subs := [y(x+h)=y(x[i+1]),y(x)=y(x[i]),y_new=y[i+1],y_old=y[i],x_new=x[i+1],x_old=x[i],
  isolate(E_def,y[i]),subs(i=i+1,isolate(E_def,y[i]))]:
  for j from 1 to 5 do:
     EOM[j] := collect(simplify(subs(Subs$2,Stencil[j]-subs(x=x[i],x[i]+h=x[i+1],_Stencil[j]
  ))),[E[i+1],E[i]]);
```

```
od;
```

$$EOM_1 := \varepsilon E_{i+1} - \varepsilon E_i - f\left(x_i, \varepsilon E_i + y\left(x_i\right)\right) h + f\left(x_i, y\left(x_i\right)\right) h = -O\left(h^2\right)$$

$$EOM_2 := \varepsilon E_{i+1} - \varepsilon E_i - f\left(x_{i+1}, \varepsilon E_{i+1} + y\left(x_{i+1}\right)\right) h + f\left(x_{i+1}, y\left(x_{i+1}\right)\right) h = -O\left(h^2\right)$$

$$EOM_3 := \varepsilon E_{i+1} - \varepsilon E_i - \frac{1}{2} f\left(x_i, \varepsilon E_i + y\left(x_i\right)\right) h - \frac{1}{2} f\left(x_{i+1}, \varepsilon E_{i+1} + y\left(x_{i+1}\right)\right) h + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h + \frac{1}{2} f\left(x_{i+1}, y\left(x_{i+1}\right)\right) h = -O\left(h^3\right)$$

$$EOM_4 := \varepsilon E_{i+1} - \varepsilon E_i - \frac{1}{2} f\left(x_i, \varepsilon E_i + y\left(x_i\right)\right) h - \frac{1}{2} hf\left(x_i + h, \varepsilon E_i + y\left(x_i\right) + f\left(x_i, \varepsilon E_i + y\left(x_i\right)\right) h\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h + \frac{1}{2} hf\left(x_{i+1}, y\left(x_i\right) + f\left(x_i, y\left(x_i\right)\right) h\right) = -O\left(h^3\right)$$

$$EOM_5 := \varepsilon E_{i+1} - \varepsilon E_i - \frac{1}{6} f\left(x_i, \varepsilon E_i + y\left(x_i\right)\right) h - \frac{1}{3} hf\left(x_i + \frac{1}{2} h, \varepsilon E_i + y\left(x_i\right) + \frac{1}{2} f\left(x_i, \varepsilon E_i + y\left(x_i\right)\right) h\right) - \frac{1}{3} hf\left(x_i + \frac{1}{2} h, \varepsilon E_i + y\left(x_i\right) + \frac{1}{2} hf\left(x_i + \frac{1}{2} h, \varepsilon E_i + y\left(x_i\right) + \frac{1}{2} f\left(x_i, \varepsilon E_i + y\left(x_i\right)\right) h\right)\right) - \frac{1}{6} hf\left(x_i + h, \varepsilon E_i + y\left(x_i\right) + hf\left(x_i + \frac{1}{2} h, \varepsilon E_i + y\left(x_i\right) + \frac{1}{2} hf\left(x_i + \frac{1}{2} h, \varepsilon E_i + y\left(x_i\right) + \frac{1}{2} f\left(x_i, \varepsilon E_i + y\left(x_i\right)\right) h\right)\right)\right) + \frac{1}{6} f\left(x_i, y\left(x_i\right)\right) h + \frac{1}{3} hf\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right) + \frac{1}{3} hf\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} hf\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right)\right) + \frac{1}{6} hf\left(x_{i+1}, y\left(x_i\right) + hf\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} hf\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right)\right)\right) = -O\left(h^5\right) \quad \textbf{(3.7}$$

In these expressions, the error terms are sometimes buried within the arguments of $f$. For example, in the first EOM (represented forward Euler), there is a term like $f\left(x_i, \varepsilon E_i + y\left(x_i\right)\right)$. To bring these out, we can expand to linear order in $\varepsilon$:

```
> for j from 1 to 5 do:
EOM_linear[j] := series((lhs-rhs)(EOM[j]),epsilon,2);
od;
```

$$EOM\_linear_1 := O\left(h^2\right) + \left(E_{i+1} - D_2(f)\left(x_i, y\left(x_i\right)\right) E_i h - E_i\right) \varepsilon + O\left(\varepsilon^2\right)$$

$$EOM\_linear_2 := O\left(h^2\right) + \left(E_{i+1} - D_2(f)\left(x_{i+1}, y\left(x_{i+1}\right)\right) E_{i+1}\, h - E_i\right) \varepsilon + O\left(\varepsilon^2\right)$$

$$EOM\_linear_3 := O\left(h^3\right) + \left(-\frac{1}{2} D_2(f)\left(x_i, y\left(x_i\right)\right) E_i\, h + E_{i+1} - E_i - \frac{1}{2} D_2(f)\left(x_{i+1}, y\left(x_{i+1}\right)\right) E_{i+1}\, h\right) \varepsilon + O\left(\varepsilon^2\right)$$

$$EOM\_linear_4 := -\frac{1}{2} h f\left(x_i + h, y\left(x_i\right) + f\left(x_i, y\left(x_i\right)\right) h\right) + \frac{1}{2} h f\left(x_{i+1}, y\left(x_i\right) + f\left(x_i, y\left(x_i\right)\right) h\right) + O\left(h^3\right) + \left(-\frac{1}{2} D_2(f)\left(x_i, y\left(x_i\right)\right) E_i\, h + E_{i+1} - E_i - \frac{1}{2} h D_2(f)\left(x_i + h, y\left(x_i\right) + f\left(x_i, y\left(x_i\right)\right) h\right) E_i \left(1 + D_2(f)\left(x_i, y\left(x_i\right)\right) h\right)\right) \varepsilon + O\left(\varepsilon^2\right)$$

$$EOM\_linear_5 := \frac{1}{6} h f\left(x_{i+1}, y\left(x_i\right) + h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right)\right)\right) + O\left(h^5\right) \quad \textbf{(3.8}$$

$$-\frac{1}{6} h f\left(x_i + h, y\left(x_i\right) + h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right)\right)\right) + \left(E_{i+1} - \frac{1}{12} h D_2(f)\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right)\right) E_i \left(4 + 2 h D_2(f)\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right) + h^2 D_2(f)\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right) D_2(f)\left(x_i, y\left(x_i\right)\right)\right) - \frac{1}{6} h D_2(f)\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right) E_i \left(2 + D_2(f)\left(x_i, y\left(x_i\right)\right) h\right) - \frac{1}{24} h D_2(f)\left(x_i + h, y\left(x_i\right) + h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right)\right)\right) E_i \left(4 + 4 h D_2(f)\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right)\right) + 2 h^2 D_2(f)\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right)\right) D_2(f)\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right) + h^3 D_2(f)\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} h f\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right)\right) D_2(f)\left(x_i + \frac{1}{2} h, y\left(x_i\right) + \frac{1}{2} f\left(x_i, y\left(x_i\right)\right) h\right) D_2(f)\left(x_i, y\left(x_i\right)\right)\right) - E_i - \frac{1}{6} D_2(f)\left(x_i, y\left(x_i\right)\right) E_i\, h\right) \varepsilon + O\left(\varepsilon^2\right)$$

We can now make a similar approximation to before: Namely, $\frac{\partial}{\partial y} f(x,y) \approx \lambda =$ constant over many iterations of the stencil. The following definitions enforce this,

```
> D[2](f) := G;
  G := (x,y) -> lambda;
  D[2](f)(x,y);
```

$$D_2(f) := G$$

$$G := (x,y) \to \lambda$$

$$\lambda \tag{3.9}$$

This greatly simpifies the equations of motion:

```
> for j from 1 to 5 do:
  EOM_linear[j] := collect(convert(subs(x[i+1]=x[i]+h,EOM_linear[j]),`+`),[E[i+1],E[i]],
  factor);
  od;
```

$$EOM\_linear_1 := \varepsilon E_{i+1} - (1+\lambda h)\,\varepsilon E_i + O(h^2) + O(\varepsilon^2)$$

$$EOM\_linear_2 := -(-1+\lambda h)\,\varepsilon E_{i+1} - \varepsilon E_i + O(h^2) + O(\varepsilon^2)$$

$$EOM\_linear_3 := -\frac{1}{2}(-2+\lambda h)\,\varepsilon E_{i+1} - \frac{1}{2}(2+\lambda h)\,\varepsilon E_i + O(h^3) + O(\varepsilon^2)$$

$$EOM\_linear_4 := \varepsilon E_{i+1} - \frac{1}{2}(2+2\lambda h+\lambda^2 h^2)\,\varepsilon E_i + O(h^3) + O(\varepsilon^2)$$

$$EOM\_linear_5 := \varepsilon E_{i+1} - \frac{1}{24}(4\lambda^3 h^3 + 24 + 24\lambda h + 12\lambda^2 h^2 + \lambda^4 h^4)\,\varepsilon E_i + O(h^5) + O(\varepsilon^2) \tag{3.10}$$

Discarding the O terms results in the same error evolution as for the test problem $y' = \lambda y$:

```
> for j from 1 to 5 do:
  Labels[j],subs(lambda=z/h,isolate(convert(EOM_linear[j],polynom),E[i+1]));
  od;
```

$$Forward\ Euler,\ E_{i+1} = (1+z)\,E_i$$

$$Backward\ Euler,\ E_{i+1} = -\frac{E_i}{-1+z}$$

$$\text{Trapezoidal}, E_{i+1} = -\frac{(2+z)\,E_i}{-2+z}$$

$$\text{Huen}, E_{i+1} = \frac{1}{2}\left(2+2z+z^2\right)E_i$$

$$\text{RK4}, E_{i+1} = \frac{1}{24}\left(4z^3+24+24z+12z^2+z^4\right)E_i \qquad \textbf{(3.11}$$

Hence, we have seen that under the assumption that the true solution is slowly varying compared to the stepsize of our stencil, the error in the numerical solution of $y'=f(x,y)$ will evolve in a manner similar to the error for $y'=\lambda y$ with $\frac{\partial}{\partial y}f \approx \lambda$. In particular, if a stencil is unstable for the test problem, it will probably be unstable for a similar nonlinear problem.