```
> restart:
> with(LinearAlgebra):
> with(CodeGeneration):
> N:=2: # Number of unknowns
> Digits:=6: # Number of digits to use for floating-point numbers
  (the numerical ODE solver will use double precision)
> with(plots):
```

Implicit part of ODE

Let's construct a linear term A*x that is stiff, with one eigenvalue of A being 0.2 and the other being lambda (an input parameter). Explicit methods will require a time step size dt~1/lambda, so for large lambda we need an implicit method

```
> eigvecs:=Matrix(N,N,[[1/3,sqrt(1-(1/3)^2)],[sqrt(1-(1/3)^2),-1/3]
  ]); # Normalized eigenvectors of matrix
```

$$eigvecs := \begin{bmatrix} \frac{1}{3} & \frac{2\sqrt{2}}{3} \\ \frac{2\sqrt{2}}{3} & \mathrm{K}\ \frac{1}{3} \end{bmatrix} \tag{1}$$

```
> simplify(MatrixInverse(eigvecs)-Transpose(eigvecs)); # Confirm
  eigenvecs is unitary
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2}$$

```
> eigvals:=lambda->Vector(N,[0.2,lambda]); # lambda is assumed to
  be a large eigenvalue, thus a stiff system
```

$$eigvals := \lambda \mapsto Vector(N, [0.2, \lambda]) \tag{3}$$

```
> A:=unapply(evalf(eigvecs.DiagonalMatrix(eigvals(lambda)).
  Transpose(eigvecs)), lambda):
> A(lambda);
```

$$\begin{bmatrix} 0.0222222 + 0.888889\,\lambda & 0.0628539\,\mathrm{K}\ 0.314269\,\lambda \\ 0.0628536\,\mathrm{K}\ 0.314269\,\lambda & 0.177777 + 0.111111\,\lambda \end{bmatrix} \tag{4}$$

```
> Eigenvectors(A(10.0)); # Confirm the eigenvecs and eigenvals are
  correct:
```

$$\begin{bmatrix} 9.99999733241764 + 0.\,I \\ 0.200002667582359 + 0.\,I \end{bmatrix}, \begin{bmatrix} 0.942809093731676 + 0.\,I & 0.333333185831919 + 0.\,I \\ \mathrm{K}\ 0.333333185831919 + 0.\,I & 0.942809093731676 + 0.\,I \end{bmatrix} \tag{5}$$

Explicit part of ODE

We will make the explicit terms be a simple quadratic function with a non-stiff Jacobian

```
> B:=x->Vector([0.2*x[1]*x[2], 0.3*x[1]^2-0.1*x[2]^2]);
```

$$B := x \mapsto Vector\left(\left[0.2\cdot x_1\cdot x_2,\ 0.3\cdot x_1^2\,\mathrm{K}\ 0.1\cdot x_2^2\right]\right) \tag{6}$$

```
> x_:=Vector([x1(t),x2(t)]);
```

$$x\_ := \begin{bmatrix} x1(t) \\ x2(t) \end{bmatrix} \tag{7}$$

Right hand side of ODE is A*x+B(x)

```
> rhs_homog:=(-A(lambda).x_+B(x_)); # Right hand side of ODEs
```

$$rhs\_homog := [[\mathrm{K}\ (0.0222222 + 0.888889\,\lambda)\,x1(t)\,\mathrm{K}\ (0.0628539\,\mathrm{K}\ 0.314269\,\lambda)\,x2(t) + 0.2\,x1(t)\,x2(t)], [\mathrm{K}\ (0.0628536\,\mathrm{K}\ 0.314269\,\lambda)\,x1(t)\,\mathrm{K}\ (0.177777 + 0.111111\,\lambda)\,x2(t) + 0.3\,x1(t)^2\ \mathrm{K}\ 0.1\,x2(t)^2]] \quad (8)$$

```
> dx_dt:=map(diff,x_,t);
```

$$dx\_dt := \begin{bmatrix} \frac{d}{dt}\,x1(t) \\ \frac{d}{dt}\,x2(t) \end{bmatrix} \quad (9)$$

```
> ODEs:=dx_dt-rhs_homog;
```

$$ODEs := \left[\left[\frac{d}{dt}\,x1(t) + (0.0222222 + 0.888889\,\lambda)\,x1(t) + (0.0628539\,\mathrm{K}\ 0.314269\,\lambda)\,x2(t)\ \mathrm{K}\ 0.2\,x1(t)\,x2(t)\right], \left[\frac{d}{dt}\,x2(t) + (0.0628536\,\mathrm{K}\ 0.314269\,\lambda)\,x1(t) + (0.177777 + 0.111111\,\lambda)\,x2(t)\ \mathrm{K}\ 0.3\,x1(t)^2 + 0.1\,x2(t)^2\right]\right] \quad (10)$$

```
> #infolevel[dsolve]:=4:
> #dsolve(convert(ODEs,set) union {x1(0)=alpha,x2(0)=beta}, {x1(t),
  x2(t)});
```

Matlab's analytical solver does not return a solution. Therefore, we will use the **method of manufactured solutions**, constructing a very simple oscillatory solution:

```
> x1_sol:=t->sin(t); x2_sol:=t->cos(t);
```

$$x1\_sol := t \mapsto \sin(t)$$

$$x2\_sol := t \mapsto \cos(t) \quad (11)$$

```
> x_sol:=Vector([x1_sol(t),x2_sol(t)]);
```

$$x\_sol := \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix} \quad (12)$$

Now compute the inhomogeneous term required to get this solution:

```
> rhs_inhomog:=eval(ODEs, {x1=x1_sol, x2=x2_sol});
```

$$rhs\_inhomog := [[\cos(t) + (0.0222222 + 0.888889\,\lambda)\sin(t) + (0.0628539\ \mathrm{K}\ 0.314269\,\lambda)\cos(t)\ \mathrm{K}\ 0.2\sin(t)\cos(t)], [\mathrm{K}\ \sin(t) + (0.0628536\,\mathrm{K}\ 0.314269\,\lambda)\sin(t) + (0.177777 + 0.111111\,\lambda)\cos(t)\ \mathrm{K}\ 0.3\sin(t)^2 + 0.1\cos(t)^2]] \quad (13)$$

Final ODE is

```
> ODE_rhs:=rhs_homog+rhs_inhomog: # rhs of ODE
> ODEs_manuf:=dx_dt-ODE_rhs:
```

```
> eval(convert(ODEs_manuf, set), {x1=x1_sol, x2=x2_sol}); # Confirm
  the manufactured solution works
```

$$\{0.\} \tag{14}$$

```
> rhs_explicit:=unapply(B(Vector([x,y]))+rhs_inhomog,(x,y,t,lambda)
  ):
> rhs_explicit(0.0,1.0,0.0,1.0); % For debugging/testing codes at
  t=0
```

$$\begin{bmatrix} 0.748585 \\ 0.288888 \end{bmatrix} \tag{15}$$

```
> Matlab(rhs_explicit(x[1],x[2],t,lambda));
cg0 = [0.2e0 * x(2) * x(1) + cos(t) + (0.222222e-1 + 0.888889e0 *
lambda) * sin(t) + (0.628539e-1 - 0.314269e0 * lambda) * cos(t) -
0.2e0 * sin(t) * cos(t) 0.3e0 * x(1) ^ 2 - 0.1e0 * x(2) ^ 2 - sin(t)
+ (0.628536e-1 - 0.314269e0 * lambda) * sin(t) + (0.177777e0 +
0.111111e0 * lambda) * cos(t) - 0.3e0 * sin(t) ^ 2 + 0.1e0 * cos(t) ^
2];
```

For fully nonlinear solvers

```
> J:=VectorCalculus[Jacobian](eval(rhs_homog,{x1(t)=x1, x2(t)=x2}),
  [x1,x2]);
```

$$J := \begin{bmatrix} \mathrm{K}\ 0.0222222\ \mathrm{K}\ 0.888889\,\lambda + 0.2\,x2 & \mathrm{K}\ 0.0628539 + 0.314269\,\lambda + 0.2\,x1 \\ \mathrm{K}\ 0.0628536 + 0.314269\,\lambda + 0.6\,x1 & \mathrm{K}\ 0.177777\ \mathrm{K}\ 0.111111\,\lambda\,\mathrm{K}\ 0.2\,x2 \end{bmatrix} \tag{16}$$

```
> J_nonlin:=VectorCalculus[Jacobian](B([x1,x2]), [x1,x2]);
```

$$J\_nonlin := \begin{bmatrix} 0.2\,x2 & 0.2\,x1 \\ 0.6\,x1 & \mathrm{K}\ 0.2\,x2 \end{bmatrix} \tag{17}$$

```
> (J_nonlin-A(lambda))-J;
```

$$\begin{bmatrix} 0. & 0. \\ 0. & 0. \end{bmatrix} \tag{18}$$

```
> Matlab(eval(J_nonlin,{x1=x[1],x2=x[2]}));
cg1 = [0.2e0 * x(2) 0.2e0 * x(1); 0.6e0 * x(1) -0.2e0 * x(2);];
```

Numerical solution in Maple

```
> lambda:=10:
> T:=evalf(6*Pi):
> A(lambda);
```

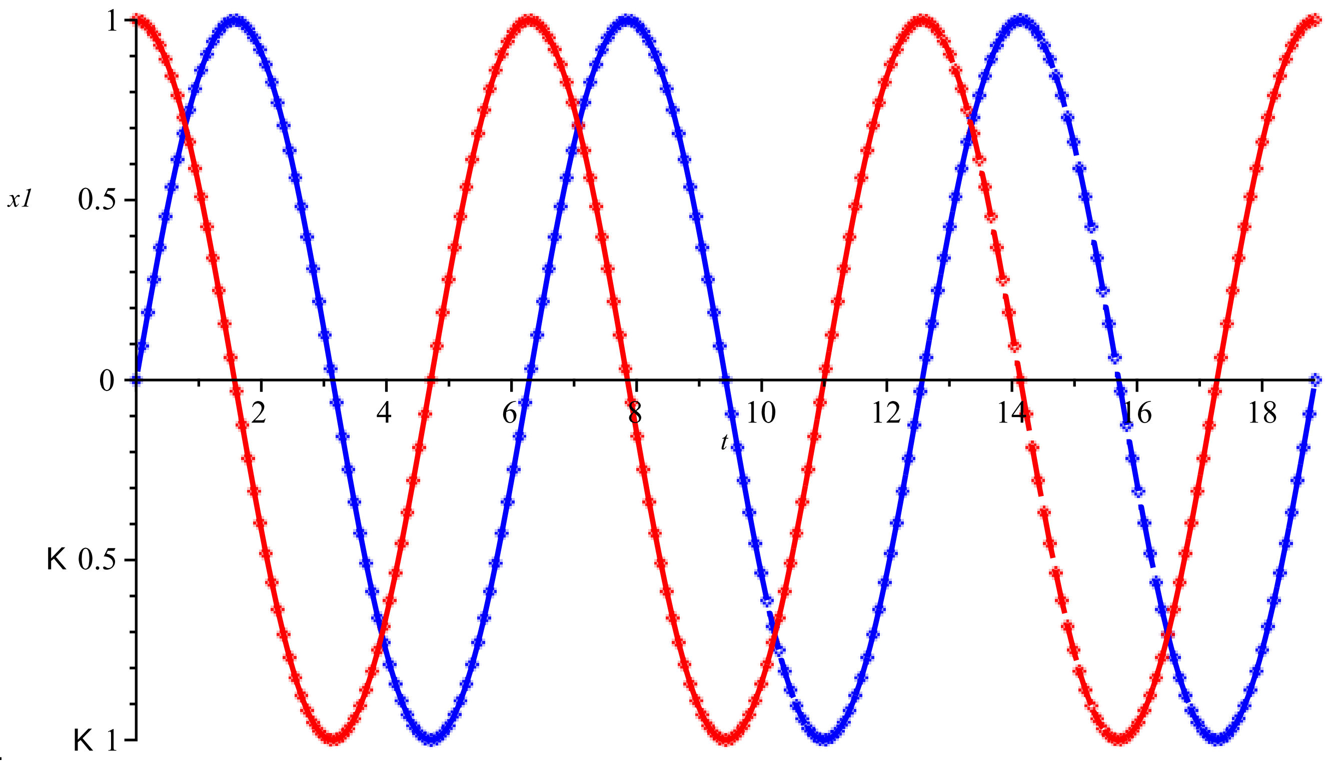
$$\begin{bmatrix} 8.91111 & \mathrm{K}\ 3.07984 \\ \mathrm{K}\ 3.07984 & 1.28889 \end{bmatrix} \tag{19}$$

Print some matrices for testing the Fortran codes:

```
> #Fortran(MatrixExponential(A(lambda)));
> #Fortran(MatrixInverse(A(lambda)));
```

Solve the ODE numerically in Maple to confirm it works

```
> num_solution:=dsolve(convert(ODEs_manuf,set) union {x1(0)=x1_sol
  (0), x2(0)=x2_sol(0)}, {x1(t),x2(t)}, type='numeric', stiff=true,
range=0..T,  relerr=1e-3, optimize=true);
```
$$num\_solution := \mathbf{proc}(x\_rosenbrock)\ \dots\ \mathbf{end\ proc} \quad (20)$$

```
> p1:=odeplot(num_solution, [t,x1(t)], color='blue', style='point')
:
> p2:=odeplot(num_solution, [t,x2(t)], color='red', style='point'):
> p3:=plot(x1_sol(t), t=0..T, color='blue', style='line',
thickness=2):
> p4:=plot(x2_sol(t), t=0..T, color='red', style='line', thickness=
2):
```
Indeed we get the correct solution [sin(t),cos(t)]
```
> display(p1,p2,p3,p4);
```


Matlab, Python, and Fortran code for A and B(x):

Fortran:
```
> explicit:=B(Vector([x[1],x[2]]))+rhs_inhomog;
```
$$explicit := \left[\begin{array}{c} 0.2\, x_1\, x_2 + 0.748585 \cos(t) + 0.911111 \sin(t) \text{ K } 0.2 \sin(t) \cos(t) \\ 0.3\, x_1^2 \text{ K } 0.1\, x_2^2 \text{ K } 1.25142 \sin(t) + 0.288888 \cos(t) \text{ K } 0.3 \sin(t)^2 + 0.1 \cos(t)^2 \end{array}\right] \quad (21)$$

```
> Fortran(explicit);
      cg(1) = 0.2D0 * x(1) * x(2) + 0.748585D0 * cos(t) + 0.911111D0
* s
     #in(t) - 0.2D0 * sin(t) * cos(t)
      cg(2) = 0.3D0 * x(1) ** 2 - 0.1D0 * x(2) ** 2 - 0.125142D1 *
sin(t
     #) + 0.288888D0 * cos(t) - 0.3D0 * sin(t) ** 2 + 0.1D0 * cos(t)
**
      #2
```

```
> A(eig);
```

$$\begin{bmatrix} 0.0222222 + 0.888889\,eig & 0.0628539\,\mathrm{K}\;\; 0.314269\,eig \\ 0.0628536\,\mathrm{K}\;\; 0.314269\,eig & 0.177777 + 0.111111\,eig \end{bmatrix} \qquad (22)$$

```
> Fortran(A(eig));
      cg0(1,1) = 0.222222D-1 + 0.888889D0 * eig
      cg0(1,2) = 0.628539D-1 - 0.314269D0 * eig
      cg0(2,1) = 0.628536D-1 - 0.314269D0 * eig
      cg0(2,2) = 0.177777D0 + 0.111111D0 * eig
```

Matlab:

```
> Matlab(explicit);
cg1 = [0.2e0 * x(1) * x(2) + 0.748585e0 * cos(t) + 0.911111e0 * sin
(t) - 0.2e0 * sin(t) * cos(t) 0.3e0 * x(1) ^ 2 - 0.1e0 * x(2) ^ 2 -
0.125142e1 * sin(t) + 0.288888e0 * cos(t) - 0.3e0 * sin(t) ^ 2 +
0.1e0 * cos(t) ^ 2];

> Matlab(A(eig));
cg2 = [0.222222e-1 + 0.888889e0 * eig 0.628539e-1 - 0.314269e0 * eig;
0.628536e-1 - 0.314269e0 * eig 0.177777e0 + 0.111111e0 * eig;];
```

Python

```
> Python(explicit);
cg3 = numpy.mat([0.2e0 * x[0] * x[1] + 0.748585e0 * math.cos(t) +
0.911111e0 * math.sin(t) - 0.2e0 * math.sin(t) * math.cos(t),0.3e0 *
x[0] ** 2 - 0.1e0 * x[1] ** 2 - 0.125142e1 * math.sin(t) + 0.288888e0
* math.cos(t) - 0.3e0 * math.sin(t) ** 2 + 0.1e0 * math.cos(t) ** 2])

> Python(A(eig));
cg4 = numpy.mat([[0.222222e-1 + 0.888889e0 * eig,0.628539e-1 -
0.314269e0 * eig],[0.628536e-1 - 0.314269e0 * eig,0.177777e0 +
0.111111e0 * eig]])
```