Matrix-Exponentials

September 7, 2017

```
In [4]: using PyPlot
```

```
INFO: Recompiling stale cache file /Users/stevenj/.julia/lib/v0.5/LaTeXStrings.ji for module LaTeXStrin
```

1 Review: Solving ODEs via eigenvectors

If we have a simple scalar ODE:

$$\frac{dx}{dt} = ax$$

then the solution is

$$x(t) = e^{at}x(0)$$

where $x(0)$ is the initial condition.
If we have an $m \times m$ system of ODEs

$$\frac{d\vec{x}}{dt} = A\vec{x}$$

we know that if $A = X\Lambda X^{-1}$ is diagonalizable with eigensolutions $A\vec{x}_k = \lambda_k\vec{x}_k$ $(k = 1, 2, \ldots, m)$, then we can write the solution as:

$$\vec{x}(t) = c_1e^{\lambda_1 t}\vec{x}_1 + c_2e^{\lambda_2 t}\vec{x}_2 + \cdots$$

where the $\vec{c}$ coefficients are determined from the initial conditions

$$\vec{x}(0) = c_1\vec{x}_1 + c_2\vec{x}_2 + \cdots$$

i.e. $\vec{c} = X^{-1}\vec{x}(0)$ where X is the matrix whose columns are the eigenvectors and $\vec{c} = (c_1, c_2, \ldots, c_m)$.

1.1 Matrix exponential, first guess:

It sure would be nice to have a formula as simple as $e^{at}x(0)$ from the scalar case. Can we **define the exponential of a matrix** so that

$$\vec{x}(t) = \underbrace{e^{At}}_{???\vec{x}(0)\,?}$$

But what is the exponential of a matrix?
We can guess at least one case. For **eigenvectors, the matrix A acts like a scalar** λ, so we should have $e^{At}\vec{x}_k = e^{\lambda_k t}\vec{x}_k$!
This turns out to be exactly correct, but let's take it a bit more slowly.

2 Writing ODE solution in matrix form

Another way of saying this is that we'd like to write the solution $x(t)$ as (some matrix) $\times \vec{x}(0)$. This will help us to understand the solution as a *linear operation on the initial condition* and manipulate it algebraically, in much the same way as writing the solution to $Ax = b$ as $x = A^{-1}b$ helps us work with matrix equations (even though we rarely compute matrix inverses explicitly in practice).

To do so, let's break down

$$\vec{x}(t) = c_1 e^{\lambda_1 t}\vec{x}_1 + c_2 e^{\lambda_2 t}\vec{x}_2 + \cdots$$

into steps.

1. Compute $\vec{c} = X^{-1}\vec{x}(0)$. That is, write the initial condition in the basis of eigenvectors. (In practice, we would solve $X\vec{c} = \vec{x}(0)$ by elimination, rather than computing X^{-1} explicitly!)
2. Multiply each component of $\vec{c}$ by $e^{\lambda t}$.
3. Multiply by X: i.e. multiply each coefficient $c_k e^{\lambda_k t}$ by $\vec{x}_k$ and add them up.

In matrix form, this becomes:

$$\vec{x}(t) = X \underbrace{\begin{pmatrix} e^{\lambda_1 t} & & & \\ & e^{\lambda_2 t} & & \\ & & \ddots & \\ & & & e^{\lambda_m t} \end{pmatrix}}_{e^{\Lambda t}} \underbrace{X^{-1}\vec{x}(0)}_{\vec{c}=} \boxed{e^{At}\vec{x}(0)}$$

where we have *defined* the "matrix exponential" of a diagonalizable matrix as:

$$e^{At} = X e^{\Lambda t} X^{-1}$$

Note that we have defined the exponential $e^{\Lambda t}$ of a *diagonal matrix* Λ to be the diagonal matrix of the $e^{\lambda t}$ values.

- Equivalently, e^{At} is the matrix with the **same eigenvectors as A but with eigenvalues λ replaced by $e^{\lambda t}$**.
- Equivalently, **for eigenvectors, A acts like a number** λ, so $e^{At}\vec{x}_k = e^{\lambda_k t}\vec{x}_k$.

2.1 Example

For example, the matrix

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

has two eigenvalues $\lambda_1 = +1$ and $\lambda_2 = -1$ (corresponding to exponentially *growing* and *decaying* solutions to $d\vec{x}/dt = A\vec{x}$, respectively). The corresponding eigenvectors are:

$$\vec{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ \vec{x}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Hence, the matrix exponential should be:

$$e^{At} = \underbrace{\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} e^{t} & \\ & e^{-t} \end{pmatrix}}_{e^{\Lambda t}} \underbrace{\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{-1}}_{X^{-1}} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} e^{t} & \\ & e^{-t} \end{pmatrix} \left[\frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\right] = \frac{1}{2}\begin{pmatrix} e^{t} & e^{-t} \\ e^{t} & -e^{-t} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} e^{t} + e^{-t} & e \\ e^{t} - e^{-t} & e \end{pmatrix}$$

In this example, e^{At} turns out to have a very nice form! In general, no one ever, ever, calculates matrix exponentials analytically like this except for toy 2×2 problems or *very* special matrices. (I will never ask you to go through this tedious algebra on an exam.)

The computer is pretty good at computing matrix exponentials, however, and in Julia this is calculated by the `expm(A*t)` function. (There is a famous paper: 19 dubious ways to compute the exponential of a matrix on techniques for this tricky problem.) Let's try it:

```
In [5]: t = 1
        [cosh(t) sinh(t)
         sinh(t) cosh(t)]

Out[5]: 2×2 Array{Float64,2}:
         1.54308  1.1752
         1.1752   1.54308

In [6]: expm([0 1; 1 0]*t)

Out[6]: 2×2 Array{Float64,2}:
         1.54308  1.1752
         1.1752   1.54308
```

Yup, it matches for $t = 1$.

What happens for larger t, say $t = 20$?

```
In [7]: t = 20
        [cosh(t) sinh(t); sinh(t) cosh(t)]

Out[7]: 2×2 Array{Float64,2}:
         2.42583e8  2.42583e8
         2.42583e8  2.42583e8

In [8]: expm([0 1; 1 0]*20)

Out[8]: 2×2 Array{Float64,2}:
         2.42583e8  2.42583e8
         2.42583e8  2.42583e8
```

For large t, the e^t exponentially growing term takes over, and $\cosh(t) \approx \sinh(t) \approx e^t/2$:

$$e^{At} = \begin{pmatrix} \cosh(t) & \sinh(t) \\ \sinh(t) & \cosh(t) \end{pmatrix} \approx \frac{e^t}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

```
In [9]: exp(20)/2 * [1 1; 1 1]

Out[9]: 2×2 Array{Float64,2}:
         2.42583e8  2.42583e8
         2.42583e8  2.42583e8
```

But we could have seen this from our eigenvector expansion too:

$$\vec{x}(t) = c_1 e^t \begin{pmatrix} 1 \\ 1 \end{pmatrix} + c_2 e^{-t} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \approx c_1 e^t \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

where c_1 is the coefficient of the initial condition: (nearly) every initial condition should give $\vec{x}(t)$ proportional to $(1, 1)$ for large t, except in the very special case where $c_1 = 0$.

In fact, since **these** two eigenvectors are an **orthogonal basis** (not by chance: we will see later that it happens because $A^T = A$), we can get c_1 just by a dot product:

$$c_1 = \frac{\vec{x}_1^T \vec{x}(0)}{\vec{x}_1^T \vec{x}_1} = \frac{\vec{x}_1^T \vec{x}(0)}{2}$$

and hence

$$\vec{x}(t) \approx c_1 e^t \vec{x}_1 = \frac{e^t}{2} \vec{x}_1 \vec{x}_1^T \vec{x}(0) = \frac{e^t}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \vec{x}(0)$$

which is the same as our approximation for e^{At} above.

3 Series definition of a matrix exponential

Just plugging in $t = 1$ above, we see that we have defined the matrix exponential by

$$e^A = X e^\Lambda X^{-1}$$

This works (for a diagonalizable matrix A, at least), but it is a bit odd. It doesn't *look* much like any definition of e^x for scalar x, and it's not clear how you would extend it to non-diagonalizable (defective) matrices.

Instead, we can **equivalently** define matrix exponentials by starting with the **Taylor series** of e^x:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots$$

It is quite natural to define e^A (for **any square** matrix A) by the **same series**:

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots + \frac{A^n}{n!} + \cdots$$

This involves only familiar matrix multiplication and addition, so it is completely unambiguous, and it converges because the $n!$ denominator grows faster than $A^n \sim \lambda^n$ for the biggest $|\lambda|$.

Let's try summing up 100 terms of this series for a random A and comparing it to both Julia's `expm` and to our formula in terms of eigenvectors:

```
In [10]: A = randn(5,5)

Out[10]: 5×5 Array{Float64,2}:
          -0.522745   0.604831   1.7945     0.355632  -0.660409
           1.48583   -1.20329    1.12371    0.573451   1.32608
          -1.41643    0.54648   -0.266285  -1.08505   -0.948708
          -0.669969  -1.17859   -0.641348  -1.49338   -0.569016
           0.607848   0.483695   0.637144   0.546879   0.2281

In [11]: expm(A)

Out[11]: 5×5 Array{Float64,2}:
           0.0310903   0.357954   0.893949  -0.370165   -0.668922
           0.165371    0.688877   1.23863   -0.116041    0.225722
          -0.458156    0.267615   0.434318  -0.428593   -0.0689023
          -0.215716   -0.663871  -1.11444    0.355304   -0.242966
           0.142495    0.430569   0.959578   0.0715271   0.926778

In [12]: series = I + A # first two terms
         term = A
         for n = 2:100
             term = term*A / n # compute A^n / n! from the previous term A^{n-1}/(n-1)!
             series = series + term
         end
         series
```

```
Out[12]: 5×5 Array{Float64,2}:
           0.0310903   0.357954   0.893949  -0.370165   -0.668922
           0.165371    0.688877   1.23863   -0.116041    0.225722
          -0.458156    0.267615   0.434318  -0.428593   -0.0689023
          -0.215716   -0.663871  -1.11444    0.355304   -0.242966
           0.142495    0.430569   0.959578   0.0715271   0.926778
```

```
In [13]: λ, X = eig(A)
         X * diagm(exp.(λ)) * inv(X)
```

```
Out[13]: 5×5 Array{Complex{Float64},2}:
          0.0310903+3.96631e-17im  ...   -0.668922+3.64067e-18im
           0.165371+3.93303e-17im         0.225722+2.76147e-17im
          -0.458156+9.31526e-18im       -0.0689023+1.11856e-18im
           -0.215716-4.2859e-17im        -0.242966-3.91105e-17im
           0.142495+3.59089e-17im         0.926778+2.83868e-17im
```

Hurray, they all match, up to roundoff errors! (Though the eigenvector method doesn't realize that the result is real, and we see tiny imaginary parts due to roundoff errors.)

But why does the eigenvector definition match the series definition? They look quite different, but they are not! We can see this in two steps:

3.1 Series definition for diagonal matrices

First, let's consider the case of e^Λ for a diagonal matrix

$$\Lambda = \begin{pmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \end{pmatrix}$$

Plugging this in, we get:

$$e^\Lambda = I+\Lambda+\frac{\Lambda^2}{2!}+\cdots = \begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix} + \begin{pmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \end{pmatrix} + \begin{pmatrix} \lambda_1^2/2! & & \\ & \lambda_2^2/2! & \\ & & \ddots \end{pmatrix} + \cdots = \begin{pmatrix} 1+\lambda_1+\lambda_1^2/2!+\cdots & \\ & 1+\lambda_2+\lambda_2^2/2! \end{pmatrix}$$

which is exactly our definition of e^Λ from the beginning!

3.2 Series definition for diagonalizable matrices

Recall that if $A = X\Lambda X^{-1}$ then $A^n = X\Lambda^n X^{-1}$. Plugging this in to the series definition, we get:

$$e^A = \underbrace{XIX^{-1}}_{I} + X\Lambda^1 X^{-1} + \frac{X\Lambda^2 X^{-1}}{2!} + \frac{X\Lambda^3 X^{-1}}{3!} + \cdots = X\left[I + \Lambda + \frac{\Lambda^2}{2!} + \cdots\right]X^{-1} = Xe^\Lambda X^{-1}$$

which exactly the "definition" we got by solving $d\vec{x}/dt = A\vec{x}$ above!

4 Matrix exponentials and derivatives

In first-year calculus, we learn that $\frac{d}{dt}e^{at} = ae^{at}$. The same thing works for matrices!

$$\boxed{\frac{d}{dt}e^{At} = Ae^{At}}$$

You can derive this in various ways. For example, you can plug e^{At} into the series definition and take the derivative term-by-term.

This is why $\vec{x}(t) = e^{At}\vec{x}(0)$ solves our ODE:

1. It satisfies $d\vec{x}/dt = A\vec{x}$, since $\frac{d}{dt}e^{At}\vec{x}(0) = Ae^{At}\vec{x}(0)$
2. It satisfies the initial condition: $e^{A\times 0}\vec{x}(0) = \vec{x}(0)$, since from the series definition we can see that $e^{A\times 0} = I$.

5 Products of matrix exponentials

In high school, you learn that $e^x e^y = e^{x+y}$. (In fact, exponentials a^x are essentially the *only* functions that have this property.)

However, this is **not** in general true for matrices:

$$\boxed{e^A e^B \neq e^{A+B}}$$

unless $AB = BA$ (unless they **commute**).

This can be seen from the series definition: if you multiply together the series for e^A and e^B, you can only re-arrange this into the series for e^{A+B} if you are allowed to re-order products of A and B. For example, the $(A+B)^2 = (A+B)(A+B)$ term gives $A^2 + AB + BA + B^2$ (not $A^2 + 2AB + B^2$!), which requires both orders BA and AB.

Let's try it:

```
In [14]: B = randn(5,5)
         expm(A) * expm(B)
```

```
Out[14]: 5×5 Array{Float64,2}:
          -3.75955    1.41223    0.171378  -1.42127    -1.10253
           0.522518   1.52749   -0.701177  -3.15371    -1.52346
          -5.13849    0.634648   0.257187   0.0583608  -0.285823
           0.93462   -1.85158    0.858997   3.44446     1.92755
           3.81969    0.642608  -1.01756   -3.15015    -1.36053
```

```
In [15]: expm(A + B)
```

```
Out[15]: 5×5 Array{Float64,2}:
           2.99024    2.35095   2.51307     -2.52295   -1.21604
           3.10479    1.56078   1.82876     -1.33485    0.0970038
          -0.820832   0.54476  -0.00778768  -0.771467  -0.179656
          -0.486885  -1.0966   -0.275423     1.01431    0.128612
           2.99489    1.13687   1.71183     -1.30094   -0.520357
```

They are not even close!

However, since A and $2A$ commute ($A \times 2A = 2A^2 = 2A \times A$), we *do* have $e^A e^{2A} = e^{3A}$:

```
In [16]: expm(A) * expm(2A)
```

```
Out[16]: 5×5 Array{Float64,2}:
          -0.16779     0.530981   0.686534  -0.444473  -0.0398259
          -0.525445    1.62803    2.72227   -1.01729    0.718321
          -0.0792024   0.676861   1.22466   -0.283008   0.509511
           0.595247   -1.87352   -3.18378    1.19056   -0.710678
          -0.512319    1.69002    2.99901   -0.922585   1.07421
```

```
In [17]: expm(3A)
```

```
Out[17]: 5×5 Array{Float64,2}:
          -0.16779     0.530981   0.686534  -0.444473  -0.0398259
          -0.525445    1.62803    2.72227   -1.01729    0.718321
          -0.0792024   0.676861   1.22466   -0.283008   0.509511
           0.595247   -1.87352   -3.18378    1.19056   -0.710678
          -0.512319    1.69002    2.99901   -0.922585   1.07421
```

```
In [18]: expm(2A) * expm(A)
```

```
Out[18]: 5×5 Array{Float64,2}:
          -0.16779     0.530981   0.686534  -0.444473  -0.0398259
          -0.525445    1.62803    2.72227   -1.01729    0.718321
          -0.0792024   0.676861   1.22466   -0.283008   0.509511
           0.595247   -1.87352   -3.18378    1.19056   -0.710678
          -0.512319    1.69002    2.99901   -0.922585   1.07421
```

5.1 Inverses of matrix exponentials

As a special case of the above, since A and $-A$ commute, we have $e^A e^{-A} = e^{A-A} = I$, so:

$$\boxed{\left(e^A\right)^{-1} = e^{-A}}$$

For example

```
In [19]: inv(expm(A))
```

```
Out[19]: 5×5 Array{Float64,2}:
           0.390707  -1.63896  -1.71987   -2.19824   -0.0229782
          -3.79365    5.059     0.62702   -0.716048  -4.11141
           2.07094   -1.21957  -0.183116   1.12123    2.07211
          -0.624538   3.85721  -0.318879   3.15695   -0.586296
          -0.393628  -1.13332   0.187338  -0.733908   0.892449
```

```
In [20]: expm(-A)
```

```
Out[20]: 5×5 Array{Float64,2}:
           0.390707  -1.63896  -1.71987   -2.19824   -0.0229782
          -3.79365    5.059     0.62702   -0.716048  -4.11141
           2.07094   -1.21957  -0.183116   1.12123    2.07211
          -0.624538   3.85721  -0.318879   3.15695   -0.586296
          -0.393628  -1.13332   0.187338  -0.733908   0.892449
```

6 Matrix exponentials as propagators

From above, we had $\vec{x}(t) = e^{At}\vec{x}(0)$ solving $d\vec{x}/dt = A\vec{x}$ given the initial condition at $t = 0$.

However, there is nothing that special about $t = 0$. We could instead have given $\vec{x}(t)$ and asked for $\vec{x}(t + \Delta t)$ and the result would have been similar:

$$\boxed{\vec{x}(t + \Delta t) = e^{A\Delta t}\vec{x(t)}} = e^{A\Delta t}e^{At}\vec{x}(0) = e^{A(t+\Delta t)}\vec{x}(0)\,.$$

Viewed in this way, the matrix $T = e^{A\Delta t}$ can be thought of as a "propagator" matrix: it takes the solution at any time t and "propagates" it forwards in time by Δt.

The *inverse* of this propagator matrix is simply $T^{-1} = e^{-A\Delta t}$, which propagates *backwards* in time by Δt.

If we multiply by this propagator matrix repeatedly, we can get $\vec{x}$ at a whole sequence of time points:

$$\vec{x}(0), \vec{x}(\Delta t), \vec{x}(2\Delta t), \ldots = \vec{x}(0), T\vec{x}(0), T^2\vec{x}(0), \ldots$$

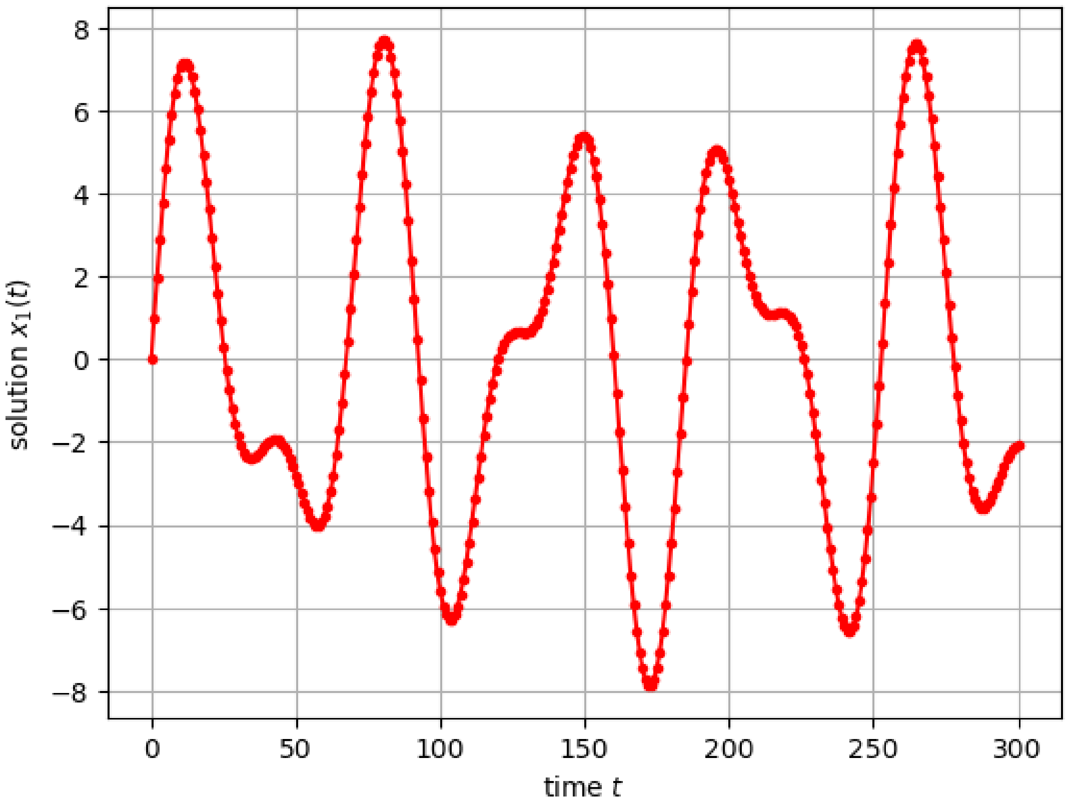
which is nice for plotting the solutions as a function of time! Let's try it for our two masses and springs example:

In [21]:
```
C = [ 0      0     1 0
      0      0     0 1
     -0.02   0.01  0 0
      0.01  -0.02  0 0 ]
Δt = 1.0
T = expm(C*Δt)  # propagator matrix

x₀ = [0.0,0,1,0]  # initial condition

# loop over 300 timesteps and keep track of x₁(t)
x = x₀
x₁ = [ x₀[1] ]
for i = 1:300
    x = T*x      # repeatedly multiply by T
    push!(x₁, x[1])  # & store current x₁(t) in the array x₁
end

plot((0:300)*Δt, x₁, "r.-")
xlabel("time \$t\$")
ylabel("solution \$x_1(t)\$")
grid()
```



(This is **not** an approximate solution. It is the *exact* solution, up to the computer's roundoff errors, at the times $t = 0, \Delta t, 2\Delta t, \ldots$. Don't confuse it with approximations like Euler's method.)

7 Key point: Stability of solutions in e[U+1D2C] vs. $\mathbf{A}^n$

It is important to compare and contrast the two cases we have studied:

- Multiplying by A^n (e.g. in **linear recurrence** equations $x_{n+1} = Ax_n$) corresponds to multiplying each eigenvector by λ^n, which:
- blows up if $|\lambda| > 1$
- decays if $|\lambda| < 1$
- oscillates if $|\lambda| = 1$
- If $\lambda = 1$ you have a steady-state vector (a stable "attractor" if the other eigenvalue have $|\lambda| < 1$).

versus

- Multiplying by e^{At} (e.g. **linear differential** equations $dx/dt = Ax$), corresponds to multiplying each eigenvector by $e^{\lambda t}$, which
- blows up if $\mathrm{Re}(\lambda) > 0$
- decays if $\mathrm{Re}(\lambda) < 0$
- oscillates if $\mathrm{Re}(\lambda) = 0$ (purely imaginary λ)
- If $\lambda = 0$ you have a steady-state solution (a stable "attractor" if the other eigenvalue have $\mathrm{Re}(\lambda) < 0$).

7.1 Relating e[U+1D2C][U+1D57] and $\mathbf{A}^n$

These two cases are **related by the propagator matrix** $T = e^{A\Delta t}$! Solving the ODE for long time, or multiplying by e^{At} for large t, corresponds to **repeatedly multiplying** by T!

What are the eigenvalues of T for a diagonalizable $A = X\Lambda X^{-1}$? Well, since

$$T = e^{A\Delta t} = Xe^{\Lambda\Delta t}X^{-1} = \begin{pmatrix} e^{\lambda_1 \Delta t} & & & \\ & e^{\lambda_2 \Delta t} & & \\ & & \ddots & \\ & & & e^{\lambda_m \Delta t} \end{pmatrix} X^{-1}$$

the eigenvalues of T are just $e^{\lambda\Delta t}$ (the equation above is precisely the diagonalization of T).

Equivalently, for an eigenvector $\vec{x}_k$ of A, $T\vec{x}_k = e^{\lambda_k \Delta t}\vec{x}_k$, so $\vec{x}_k$ is **also an eigenvector** of T with eigenvalue $e^{\lambda_k \Delta t}$. Let's check:

```
In [ ]: eigvals(expm(A*Δt))
```

```
In [ ]: λ = eigvals(A)
        exp.(λ * Δt)
```

Yup, they match (although the order is different: Julia gives the eigenvalues in a somewhat "random" order).

What does this mean for stability of the solutions?

For example, if A has an real eigenvalue with $\lambda < 0$, a decaying solution, then T has an eigenvalue $e^{\lambda\Delta t} < 1$, which is also decaying when you multiply by T repeatedly!

It is easy to verify that going from $\lambda \to e^{\lambda}$ turns the **conditions for growing/decaying ODE (e[U+1D2C][U+1D57]) solutions into the rules for growing/decaying $\mathbf{A}^n$ solutions!**.