Idag: 1. Mer om system av ODE
- stabilitet
- komplexa egenvärden

2. Numerisk lösning av egenvärdesproblem (5.8)

Linjärt system av ODE:

$$\begin{cases} x'(t) = Ax(t), & t>0 \\ x(0) = b \end{cases}$$ (kopplat system)

$A = PDP^{-1}$ (se F14.)

Koordinatbyte: $x = Py$, $b = Pc$

$$\begin{cases} y'(t) = Dy \\ y(0) = c \end{cases}$$ diagonalt system

$y_k'(t) = \lambda_k y_k(t) \Rightarrow y_k(t) = c_k e^{\lambda_k t}$

$x(t) = Py(t) = \sum_{k=1}^{n} c_k e^{\lambda_k t} v_k$

Formeln ger förståelse.

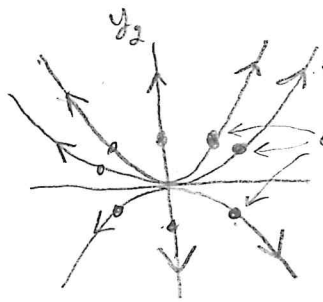
Ex ($n=2$) $x(t) = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2$

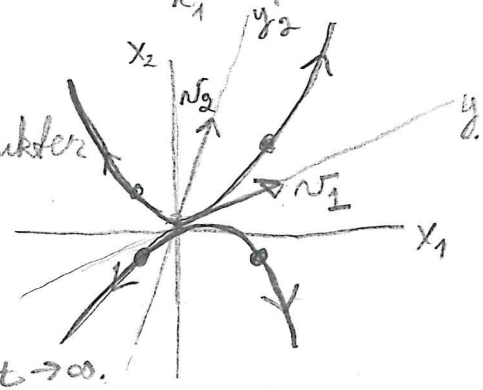
Fall 1: Båda $\lambda_1 > 0$ och $\lambda_2 > 0$.

Tex $\lambda_1 = 1$, $\lambda_2 = 2$.

$x(t) = c_1 e^t v_1 + c_2 e^{2t} v_2$

$y_2(t) = c_2 e^{2t} = \frac{c_2}{c_1^2}(c_1 e^t)^2 = \frac{c_2}{c_1^2} y_1(t)^2$ ($c_1 \neq 0$)

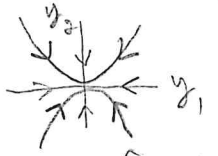


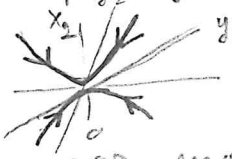


Alla lösningar $\to \infty$ då $t \to \infty$.
Instabilt system.

Fall 2: Båda $\lambda_1 < 0$, $\lambda_2 < 0$. Ex $\lambda_1 = -1$, $\lambda_2 = -2$.

$x(t) = c_1 e^{-t} v_1 + c_2 e^{-2t} v_2$, $y_2 = c y_1^2$

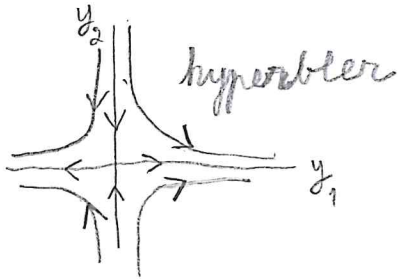


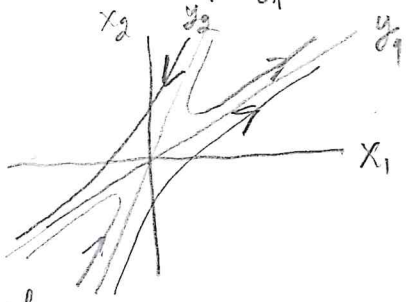


Alla lösningar går mot 0, då $t \to \infty$.
Stabilt system

Fall 3: En $\lambda_1>0$ och en $\lambda_2<0$. T.ex. $\lambda_1=1$, $\lambda_2=-2$.

$$y_2(t) = c_2 e^{-2t} = \frac{c_2}{c_1^2}\left(\frac{c_1}{e^t}\right)^2 = \frac{c_2}{c_1^2}\,\frac{1}{y_1(t)^2}$$





Hyperbolisk system
Instabilt

Ex Svängning

$$\begin{cases} u''(t)+\omega^2 u(t)=0, & t>0 \\ u(0)=u_0,\ u'(0)=u_1 \end{cases} \quad (\omega>0)$$

$$\begin{cases} x_1 = u \\ x_2 = u' \end{cases} \qquad \begin{cases} x_1' = x_2 \\ x_2' = -\omega^2 x_1 \end{cases}$$

$x' = Ax$ med $A=\begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix}$

$$\det(A-\lambda I)=\begin{vmatrix} -\lambda & 1 \\ -\omega^2 & -\lambda \end{vmatrix} = \lambda^2+\omega^2 = 0, \quad \lambda_1 = i\omega,\ \lambda_2 = -i\omega$$

Med $\lambda = i\omega$: $\begin{bmatrix} -i\omega & 1 \\ -\omega^2 & -i\omega \end{bmatrix} \overset{i\omega}{\leftarrow} \sim \begin{bmatrix} -i\omega & 1 \\ 0 & 0 \end{bmatrix} \overset{i/\omega}{\sim} \begin{bmatrix} 1 & i/\omega \\ 0 & 0 \end{bmatrix}$

$x_2 = s$ fri, $x = s\begin{bmatrix} -i/\omega \\ 1 \end{bmatrix} = \left(\frac{-i}{\omega}s\right)\begin{bmatrix} 1 \\ i\omega \end{bmatrix}$

Tag $v_1 = \begin{bmatrix} 1 \\ i\omega \end{bmatrix}$.

Med $\lambda = -i\omega$, på liknande sätt:

$$v_2 = \begin{bmatrix} 1 \\ -i\omega \end{bmatrix}.$$

$$P = [v_1, v_2] = \begin{bmatrix} 1 & 1 \\ i\omega & -i\omega \end{bmatrix}, \quad \det(P) = -2i\omega \neq 0$$

$$D = \begin{bmatrix} i\omega & 0 \\ 0 & -i\omega \end{bmatrix}, \quad P^{-1} = \frac{1}{-2i\omega}\begin{bmatrix} -i\omega & -1 \\ -i\omega & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{-i}{2\omega} \\ \frac{1}{2} & \frac{i}{2\omega} \end{bmatrix}$$

$$\begin{aligned} X(t) &= c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2 = \\ &= c_1 e^{i\omega t}\begin{bmatrix} 1 \\ i\omega \end{bmatrix} + c_2 e^{-i\omega t}\begin{bmatrix} 1 \\ -i\omega \end{bmatrix} \\ &= c_1\big(\cos(\omega t)+i\sin(\omega t)\big)\begin{bmatrix} 1 \\ i\omega \end{bmatrix} \\ &\quad + c_2\big(\cos(\omega t)-i\sin(\omega t)\big)\begin{bmatrix} 1 \\ -i\omega \end{bmatrix} \end{aligned}$$

$$\begin{aligned} x_1(t) &= (c_1+c_2)\cos(\omega t) + (c_1-c_2)i\sin(\omega t) \\ &= a\cos(\omega t) + b\sin(\omega t) \end{aligned}$$

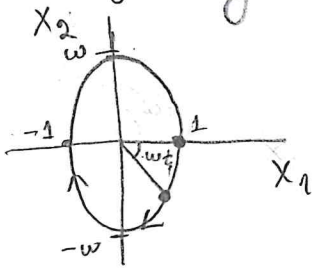
$$\begin{aligned} x_2(t) &= -(c_1+c_2)\omega\sin(\omega t) + (c_1-c_2)i\omega\cos(\omega t) \\ &= \omega\big(-a\sin(\omega t)+b\cos(\omega t)\big) \end{aligned}$$

där $a = c_1+c_2$, $b = i(c_1-c_2)$ väljs reella.

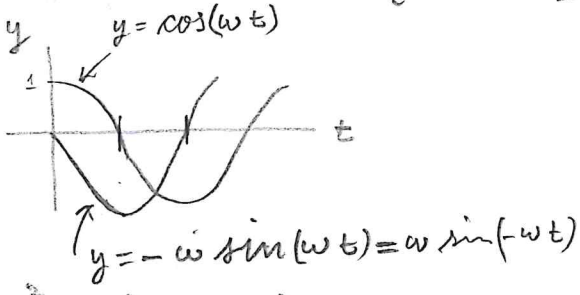
obs: $x_2(t) = x_1'(t)$ som det ska vara.

$u(t) = a\cos(\omega t) + b\sin(\omega t)$

Svängning! T. ex. $a=1$, $b=0$, $X(t) = \begin{bmatrix} \cos(\omega t) \\ -\omega\sin(\omega t) \end{bmatrix}$



medurs på ellips



$x_1^2 + \frac{1}{\omega^2}x_2^2 = \cos^2(\omega t) + \sin^2(\omega t) = 1$

Mer allmänt: $\lambda = \alpha + i\omega$.

$e^{\lambda t} = e^{\alpha t}e^{i\omega t} = e^{\alpha t}(\cos(\omega t) + i\sin(\omega t))$

$|e^{\lambda t}| = e^{\alpha t} \to 0$ om $\alpha < 0$.

Stabilt system om alla $\operatorname{Re}\lambda_k = \alpha_k < 0$.

Instabilt annars.

Numerisk lösning av egenvärdesproblem.

Det är opraktiskt (omöjligt) lösa $\det(A-\lambda I) = 0$ för det är polynomekvation av grad n.

Vad göra? Approximera.

Idé: beräkna en följd av vektorer $x_1, x_2, \ldots = (x_k)_{k=1}^{\infty}$ som konvergerar mot en egenvektor v med

$Av = \lambda v$.

Dvs $x_k \to v$, $k \to \infty$,

och vi får en approximation av egenvärdet genom Rayleigh-kvoten:

$$\frac{x_k^T A x_k}{x_k^T x_k} \to \frac{v^T A v}{v^T v} = \frac{v^T(\lambda v)}{v^T v} = \lambda$$

Obs: $v^T v = |v|^2$

Den enklaste algoritmen är potensmetoden.

Tag godtyckligt x_0 och beräkna

$x_k = A^k x_0$, dvs $x_1 = Ax$, ..., $x_k = Ax_{k-1}$.

Antag $A \in \mathbb{R}^{n\times n}$ har bas av egenvektorer $\{v_1, \dots, v_n\}$. Då fås

$$x_0 = c_1 v_1 + \dots + c_n v_n = \sum_{j=1}^{n} c_j v_j$$

och

$$x_k = c_1 \lambda_1^k v_1 + \dots + c_n \lambda_n^k v_n .$$

Om nu

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$$

dvs λ_1 är dominant egenvärde,

så får vi

$$\frac{1}{\lambda_1^k} x_k = c_1 v_1 + c_2 \underbrace{\left(\frac{\lambda_2}{\lambda_1}\right)^k}_{\to 0} v_2 + \dots + c_n \underbrace{\left(\frac{\lambda_n}{\lambda_1}\right)^k}_{\to 0} v_n \to c_1 v_1$$

Alltså: $\frac{1}{\lambda_1^k} x_k \to c_1 v_1$ en egenvektor till λ_1.

(Måste ha $c_1 \neq 0$, annars ta ny startvektor.)

Den konvergenta iterationen är

$$y_1 = \frac{1}{\lambda_1} A x_0$$

$$y_k = \frac{1}{\lambda_1} A y_{k-1} \qquad \left(y_k = \frac{1}{\lambda_1^k} x_k \right)$$

dvs multiplicera med A och skala med $\frac{1}{\lambda_1}$.

Men λ_1 är okänd, så skala på något annat sätt.

T. ex. normera:

$$\begin{cases} x_k = A y_{k-1} \\ y_k = \pm \frac{1}{|x_k|} x_k \end{cases}$$

där tecknet väljs så att y_k och y_{k-1} pekar åt samma håll.

Se datorövning.

I boken normerar man så att största komponenten i y_k blir $= 1$.

Hur beräkna andra egenvärden än det dominanta?

Skifta och invertera.

$$B = A - \alpha I \qquad (\text{skifta}, \ \alpha \in \mathbb{R})$$

$$B^{-1} = (A - \alpha I)^{-1} \qquad (\text{invertera})$$

B och B^{-1} har samma egenvektorer som A, ty

$$Av = \lambda v$$

$$\Rightarrow Bv = Av - \alpha v = (\lambda - \alpha) v$$

$$\Rightarrow Bv = (\lambda - \alpha) v$$

$$\Rightarrow v = (\lambda - \alpha) B^{-1} v$$

$$\Rightarrow B^{-1} v = (\lambda - \alpha)^{-1} v$$

Alltså: B^{-1} har egenvärde $(\lambda - \alpha)^{-1}$ med samma egenvektor v.

Om α nära λ så blir $(\lambda - \alpha)^{-1}$ väldigt dominant egenvärde för B^{-1}.

Detta leder till inversa potensmetoden:

Gissa en approximation $\alpha \approx \lambda$.
Kör potensmetoden på
$$B^{-1} = (A - \alpha I)^{-1}.$$

Se datorövning.

Nackdel: ger bara ett egenpar λ, v åt gången.

Idén kan förfinas så att man beräknar flera (alla) egenvärden på en gång.
Matlabs eig( ) bygger på QR-metoden, se slutet av F15.

Ex Lösa polynomekvation som egenvärdesproblem.

Vi har bestämt egenvärden som lösningarna till polynomekvationen
$$\det(A - \lambda I) = 0$$
(Algebrans fundamentalsats!)
Detta är viktig teori men omöjligt som beräkningsmetod.
(Beräkna alla rötter med Newtons metod!!)
Men nu har vi effektiva numeriska metoder som beräknar alla egenvärde till en given matris.

Låt $p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1} + x^n$ $(c_n = 1)$
Om $p(\lambda) = 0$ så är λ egenvärde till matrisen

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & 1 \\ -c_0 & \cdots & \cdots & -c_{n-2} & -c_{n-1} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

med egenvektor $v = \begin{bmatrix} 1 \\ \lambda \\ \vdots \\ \lambda^{n-1} \end{bmatrix} \in \mathbb{R}^n$.

Bevis: $Av = \cdots = \lambda v$

Matlab: eig(A) ger alla rötter till $p(x)$.

T.ex. $p(x) = (x-2)(x-3)(x-5)$
$= x^3 - 10x^2 + 31x - 30$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 30 & -31 & 10 \end{bmatrix}$$

eig(A): $D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix}$, $P = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 5 \\ 4 & 9 & 25 \end{bmatrix} = \begin{bmatrix} 2^0 & 3^0 & 5^0 \\ 2^1 & 3^1 & 5^1 \\ 2^2 & 3^2 & 5^2 \end{bmatrix}$

```
%% Konstruera matris A som är diagonaliserbar
clear all

P = [1 1 1; 1 2 -3; 4 5 6];   % Hitta på egenvektorer
D = [5 0 0; 0 1  0; 0 0 3];   % Hitta på egenvärden
D = [-5 0 0; 0 1  0; 0 0 3];   % Hitta på egenvärden, negativt dominant
% D = [5 0 0; 0 1 0;0 0 -5];   % Hitta på egenvärden, inget dominant egv

A = P*D/P                     % P*D*inv(P)

%% Räkna ut egenvektorer och egenvärden med MATLABs metod:
[V, E] = eig(A)

%% Potensiteration
x = rand(3, 1)  % Slumpmässig startvektor

%% Upprepa detta till konvergens
x = A*x;
x = x/norm(x)               % normera så att x inte växer eller avtar
                            % obs: x flippar om lambda<0

%% lambda = (x'*A*x)/(x'*x)    Rayleigh-kvoten
lambda = x'*A*x              % Rayleigh-kvoten, x är normerad
```

```
%% Konstruera matris A som är diagonaliserbar
clear all

P = [1 1 1; 1 2 -3;4 5 6];   % Hitta på egenvektorer
D = [5 0 0; 0 1 0;0 0 3];    % Hitta på egenvärden

A = P*D/P                     % P*D*inv(P)

%% Räkna ut egenvektorer och egenvärden med MATLABs metod:
[V, E] = eig(A)

%% Invers potensiteration
x = rand(3, 1)  % Slumpmässig startvektor

alpha = 2.5     % Nära 3
% alpha = 1.5    % Nära 1
I = eye(size(A));

%% Upprepa detta till konvergens
x = (A - alpha*I) \ x;      % Mult med inversen  x=(A-alpha*I)^(-1)*x
x = x/norm(x)               % Normera

%% lambda = (x'*A*x)/(x'*x)    Rayleigh-kvoten
lambda = x'*A*x              % Rayleigh-kvoten, x är normerad
```

```
%% Konstruera A
clear all

P = [1 1 1; 1 2 -3;4 5 6];  % Hitta på egenvektorer
D = [5 0 0; 0 1 0;0 0 3];   % Hitta på egenvärden

A = P*D/P   % P*D*inv(P)

%% Räkna ut egenvektorer och egenvärden med MATLABs inbyggda metod:
[V, E] = eig(A)

%% QR-algoritmen
X=A;

%% Upprepa detta till konvergens
[Q, R] = qr(X)          % QR-faktorisering, X=Q*R, Q'*Q=I, R triangulär
X = R*Q                 % X är triangulär och similär med A
                        % X_{k+1} = Q_k^{-1} * X_k * Q_k
```