Algoritmus metody konečných prvků

- Základní algoritmus
- Možnosti automatizace
- Okrajové podmínky, zatížení...

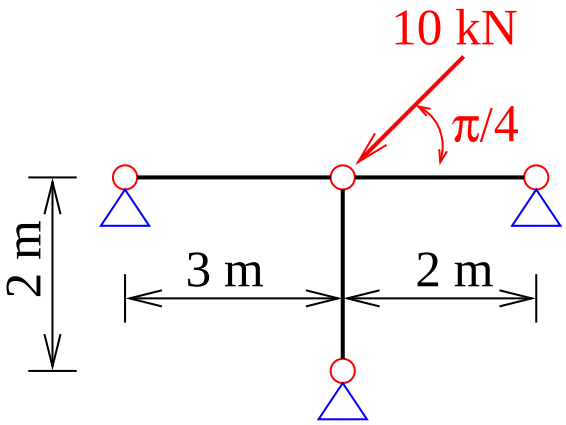
Základní algoritmus

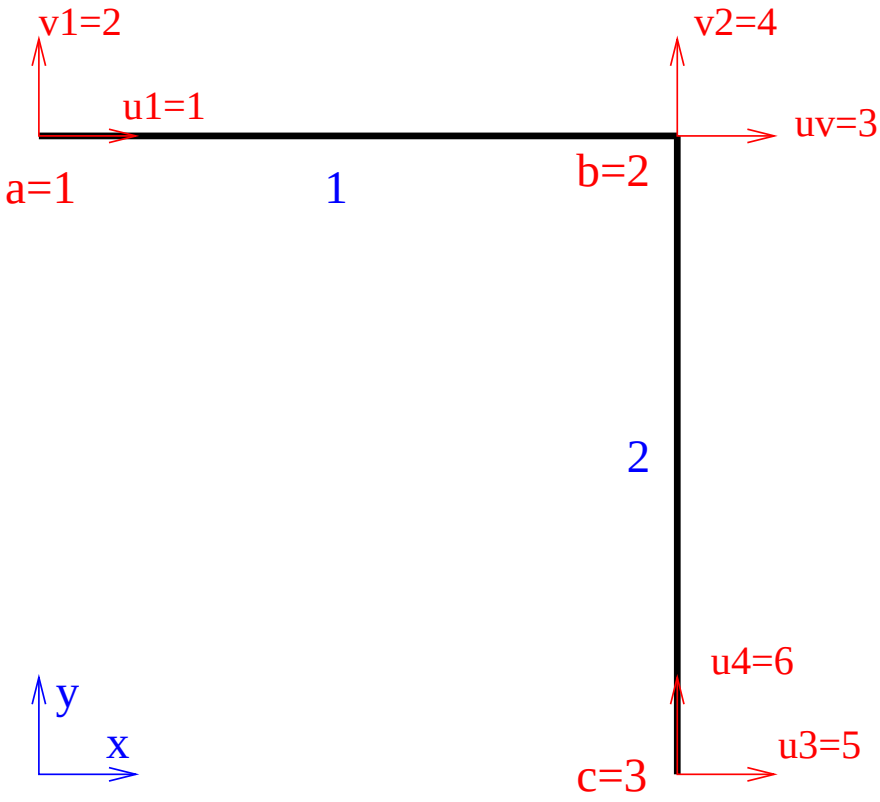
1. Zpracování zadání
2. Sestavení matice tuhosti konstrukce (pro všechny prvky):
   - Matice tuhosti prvku
   - Transformace do globálních souřadnic
   - Lokalizace do matice tuhosti konstrukce
3. Globální vektor zatížení
4. Okrajové podmínky
5. Výpočet deformací
6. Výpočet výsledků (pro všechny prvky):
   - Převod deformací do vektorů prvků (a transformace)
   - Výpočet poměrných deformací, napětím, vnitřních sil
7. Výpočet reakcí

Pozn.: uvedený algoritmus platný například pro *lineární statický výpočet* je ne bezdůvodně shodný s postupem obecné deformační metody (ODM).

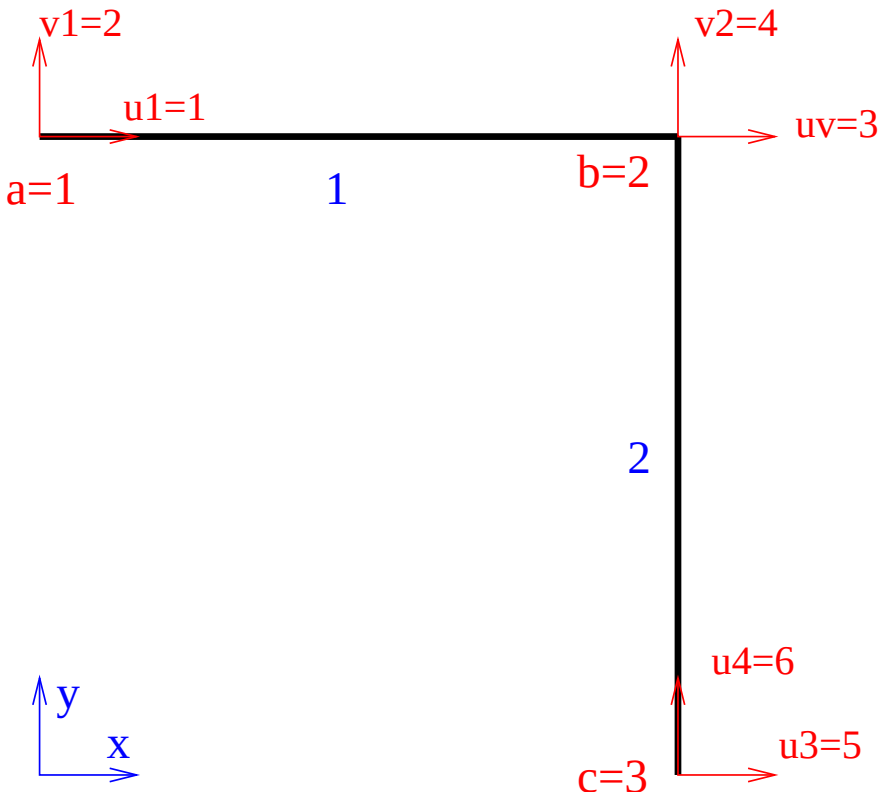
Ilustrace programu

Program pro řešení příhradových konstrukcí.





Zadání: typ úlohy, materiál



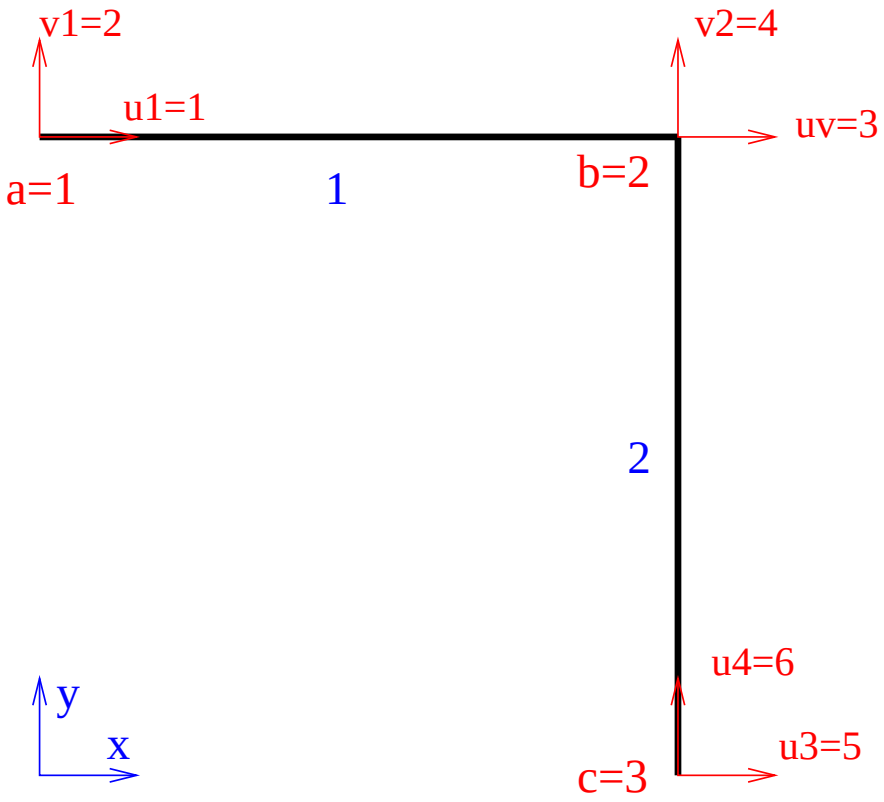
Počet stupňů volnosti (*ndof*) a počet uzlů na prvku (*puzlu*):

```
ndof=2
puzlu=2
```

Modul pružnosti (*E*) a plocha prvku(*A*):

```
E=20e9
A=0.01
```

Zadání: geometrie



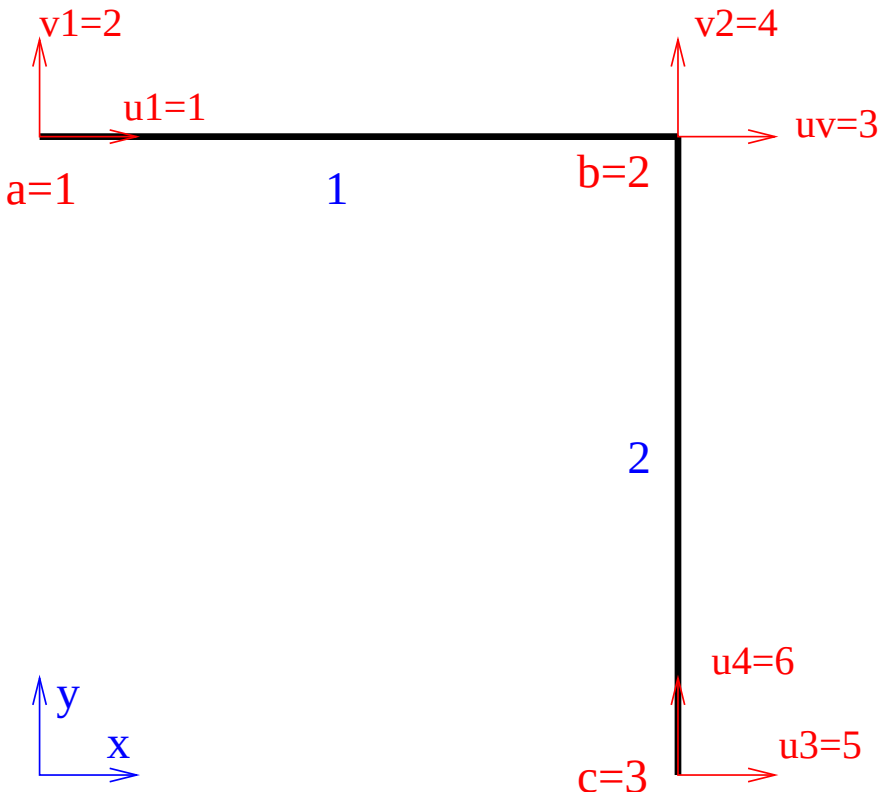
Souřadnice uzlů (*x,y*, co řádek to prvek):

```
uzly=[ 0 2 ;
       3 2 ;
       3 0 ]
```

Konečné prvky (čísla uzlů na prvku – pořadová čísla řádků v matici *uzly*).

```
pruty=[ 1 2 ;
        3 2 ]
```

Zadání: podpory a zatížení

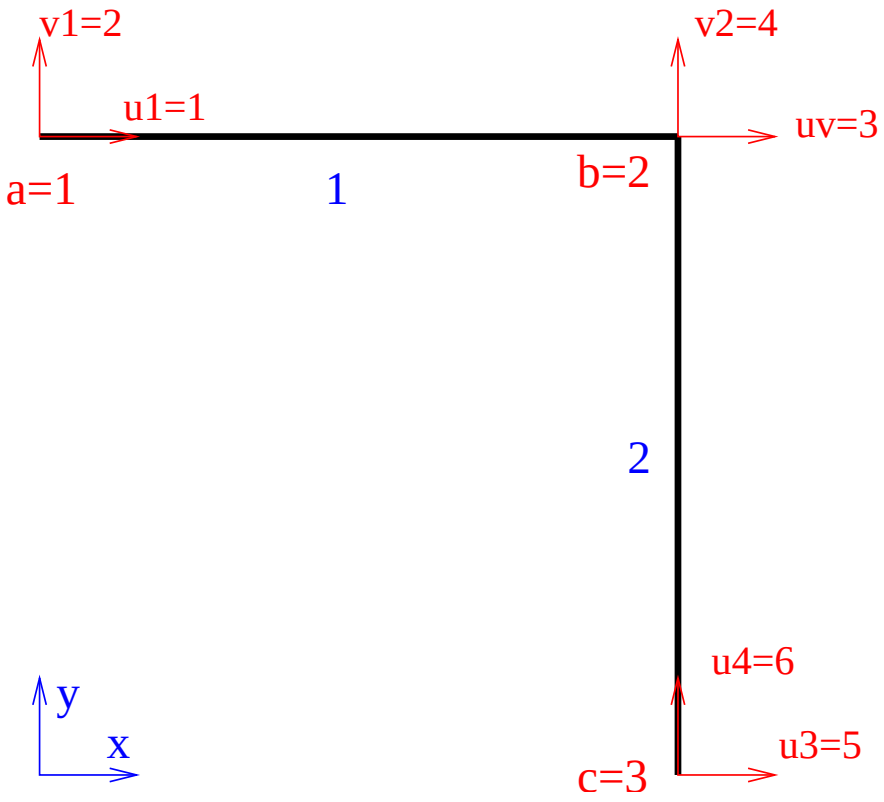



```
podpory=[
1 1 0 ;
1 2 0  ;
3 1 0 ;
3 2 0 ]

sily=[
2 1 -2000 ;
2 2 -3000 ]
```

Pomůcka: číslo uzlu, směr (1=u, 2=v), velikost.

Výpočet velikosti polí



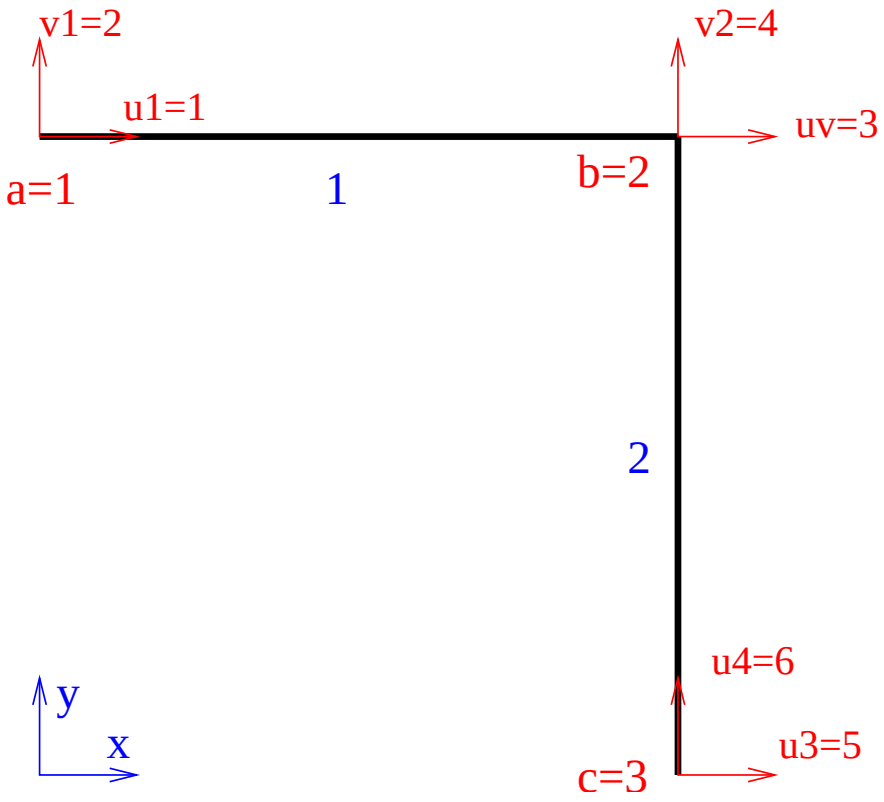
Zadané veličiny:

```
nuzlu=size(uzly,1)
nprutu=size(pruty,1)
npodpor=size(podpory,1)
nsil=size(sily,1)
```

Velikost úlohy:

```
velikost = nuzlu*ndof
```

Kódová čísla (1)



Prut 1 (uzly a=1,b=2):

$$[u1, v1, u2, v2] = [1, 2, 3, 4]$$

Prut 2 (uzly c=3,b=2):

$$[u3, v3, u2, v2] = [5, 6, 3, 4]$$

Kódová čísla (2)

Sestavíme kódová čísla:

```
kcis=zeros(nprutu,ndof*puzlu);
for i=1:nprutu
  for j=1:puzlu
    for k=1:ndof
      kcis(i, ((j-1)*ndof+k))=(pruty(i,j)-1)*ndof+k ;
    end
  end
end
```

Kódová čísla (3)

Jak to funguje:

- Pro každý prut si připravíme jeden řádek
- Zjistíme číslo prvního uzlu (např. 7)
- Každý uzel má $ndof = 2$ stupně volnosti (u_i, v_i), tedy kódová čísla budou:
  - (Uzel 1: $u_1 = 1 \times 2 - 1 = 1$, $v_1 = 1 \times 2 = 2$)
  - **Uzel** 7: $u_7 = 7 \times 2 - 1 = 13$, $v_7 = 7 \times 2 = 14$
- Zjistíme číslo druhého uzlu (např. 8)
- Vypočítáme jeho kódová čísla:
  - **Uzel** 8: $u_8 = 8 \times 2 - 1 = 15$, $v_8 = 8 \times 2 = 16$
- **Výsledek pro oba uzly** zapíšeme do vektoru - řádku: $[13, 14, 15, 16]$
- Postup provedeme pro všechny pruty

Až sestavíme matici tuhosti prutu, tak uvedený řádek bude popisovat polohu jednotlivých prvků v globální matici tuhosti (tj. například členy **1. řádku matice tuhosti prutu** budou ležet ve *13. řádku matice tuhosti konstrukce*).

Podobně později vypočítáme i kódová čísla pro podpory a polohy sil v uzlech.

Výpočet: nulování K, U, F

```
K = zeros(velikost);
u = zeros(velikost,1);
F = zeros(velikost,1);
```

Výpočet: sestavení K

```
for i=1:nprutu
  % TADY BUDE sestaveni lokalni matice tuhosti:

  % TADY BUDE transformace do glob. souradnic:

  % TADY BUDE samotna lokalizace:
end
```

Výpočet: sestavení lokální Ke

```
dx = (uzly(pruty(i,1),1)-uzly(pruty(i,2),1));
dy = (uzly(pruty(i,1),2)-uzly(pruty(i,2),2));
L = sqrt(dx^2 + dy^2);

S=[1 0 ; 1 L];
B=[0 1];
D=[E];
Si=inv(S);
Kel=A*L*Si'*B'*D*B*Si;
Keg=[Kel(1,1) 0 Kel(1,2) 0 ; 0 0 0 0 ;
    Kel(2,1) 0 Kel(2,2) 0 0 0 0 0 ];
```

Výpočet: transformace Ke do globálních souřadnic

```
% Transformace:
s = dy/L;
c = dx/L;

T = [ c s 0 0 ; -s c 0 0 ;
      0 0 c s ; 0 0 -s c ] ;

 Ke = T' * Keg * T;
```

Výpočet: lokalizace do K

```
for j=1:(puzlu*ndof)
for k=1:(puzlu*ndof)
K(kcis(i,j),kcis(i,k))=K(kcis(i,j),kcis(i,k))+Ke(j,k);
end
end
```

Výpočet: podpory (1)

```
for i=1:npodpor
  iuz=podpory(i,1);
  ismer=podpory(i,2);
  pos = (ndof*(iuz-1))+ismer;
  u(pos) = u(pos)+podpory(i,3);
  for j=1:velikost
    K(pos,j) = 0.0 ;
    K(j,pos) = 0.0 ;
  end
  K(pos,pos) = 1.0 ;
end
```

Pozn.: výpočet `pos = (ndof*(iuz-1))+ismer` odpovídá postupu při výpočtu kódových čísel (jen s jinak označenými proměnnými...).

Výpočet: podpory (2)

Uvedený algoritmus funguje jen pro pevné podpory (s nulovým posunutím).

Pro předepsaná posunutí v řádku i (popuštění podpor) by bylo ještě třeba:

- doplnit do vektoru *u* příslušnou hodnotu posunutí u_i
- hodnoty v i-tém sloupci matice $\mathbf{K}$ vynásobit hodnoutou u_i a odečíst od vektoru $\mathbf{F}$
- na diagonálu matice $\mathbf{K}$ v i-tém řádku dosadit 1
- do vektoru $\mathbf{F}$ v i-tém řádku dosadit u_i

Detaily viz příslušný postup v obecné deformační metodě (pozor – ten, kde se *nepracuje* s primárními vektory, ale hodnoty se zadávají do vektoru deformací konstrukce).

Výpočet: zatížení

Předpokládáme jen síly v uzlech:

```
for i=1:nsil
  iuz=sily(i,1);
  ismer=sily(i,2);
  pos = (ndof*(iuz-1))+ismer;
  F(pos) = F(pos)+sily(i,3);
end
```

Pozn.: výpočet `pos = (ndof*(iuz-1))+ismer` odpovídá postupu při výpočtu kódových čísel (a je identický s výpočtem použitým u podpor).

Výpočet: soustava rovnic

Vyřešíme soustavu rovnic $\mathbf{K} \times \mathbf{u} = \mathbf{F}$ (je-li vše dobře, tak jsou lineární):

```
u=K\F
```

Vektor $\mathbf{u}$ obsahuje hledaná posunutí.

Poznámky:

- Operátor ,,zpětné lomítko" označuje operaci řešení soustavy rovnic.
- Zápis `u = inv(K) * F`, oblíbený u uživatelů tabulkových kalkulátorů, je u větších úloh naprosto nevhodný (nutí program vyřešit inverzní matici, tj. soustavu o n rovnicích řešit n-krát!),

Výsledky: výpočet na jednotlivých prvcích

```
for i=1:nprutu
ue  = zeros(puzlu*ndof,1);
uel = zeros(puzlu*ndof,1);

% TADY BUDE ziskani lokalnich vektoru deformaci:
% TADY BUDE transformace:
% TADY BUDE matice tuhosti (jen lokalni):
  % TADY BUDE vysledeK - sily v prutech:
end
```

Výsledky: lokální vektory deformací

```
% Ziskani lokalnich vektoru deformaci:

for j=1:(puzlu*ndof)
ue(j) = u(kcis(i,j));
end
```

Výsledky: transformace lokálních vektorů deformací

Vektory převedeme do lokálního systému souřadnic:

```
dx2 = (uzly(pruty(i,1),1)-uzly(pruty(i,2),1))^2;
dy2 = (uzly(pruty(i,1),2)-uzly(pruty(i,2),2))^2;
L = sqrt(dx2 + dy2);

s = sqrt(dy2)/L;
c = sqrt(dx2)/L;
  T = [ c s 0 0 ; -s c 0 0 ;
        0 0 c s ; 0 0 -s c ] ;
  uel = T * ue;
```

Výsledky: lokální matice tuhosti

```
S=[1 0 ; 1 L];
B=[0 1];
D=[E];
Si=inv(S);
Kel=A*L*Si'*B'*D*B*Si;
Keg=[Kel(1,1) 0 Kel(1,2) 0 ;
     0 0 0 0 ;
     Kel(2,1) 0 Kel(2,2) 0
     0 0 0 0 ];
```

Výsledky: koncové síly na prutech

```
Fe = Keg * uel
```

Poznámky:

- Síly jsou **v globálním** systému souřadnic
- U prutových konečnýc hprvků je lze využít stejně jako v obecné deformační metodě.
- U 2D a 3D prvků se příliš nevyužívají (vypočítáme přímo σ, ε, viz příště).
- Vždy však poslouží při výpočtu reakcí v uzlu - sečteme příspěvky (ve směru podpory) ze všech prutů připojených do daného uzlu.

Doplnění: matice tuhosti K (1)

Vlasnosti matice tuhosti konstrukce (v úlohách lineární pružnosti):

- **pozitivně definitní** (jen pokud je konstrukce řádně podepřená!),
- **symetrická podle hlavní diagonály** (vyplývá např. z Bettiho věty),
- **řídká**, zpravidla pásová (obsahuje velmi málo nenulových členů).

Uvedených vlastností využívají efektivní řešiče soustav rovnic (obvykle se využívají metody na bázi **Gaussovy eliminace** v úpravách pro řídké matice nebo varianty **metody sdružených gradientů**).

P.S. Na dalším snímku zobrazená matice není úplně idaální – proč? A jak k tomu asi došlo?

Doplnění: matice tuhosti K (2)