Zpravodaj Československého sdružení uživatelů TeXu

Oldřich Ulrych

Zkušenosti s METAFONTem

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 2 (1992), No. 2, 60–80

Persistent URL: http://dml.cz/dmlcz/149622

Zkušenosti s METAFONTem

Důvody pro tento příspěvek

Tento článek v jistém smyslu navazuje na článek zkušeného METAFONTisty Karla Horáka (TEXbulletin 3 (1991)), a chceme se v něm podělit s laskavými čtenáři o naše zkušenosti s používáním METAFONTu pro kreslení obrázků, se začleňováním obrázků do textu v TEXu, obtékání obrázků... V tomto úvodu bychom mohli pokračovat dál dost dlouho. Pokud si však chceme nalít čistého vína (a nasypat popel na hlavu), musíme říct toto:

a) Při kreslení obrázků s použitím METAFONTu jen velice těžko hledáme syntaxi jednotlivých příkazu METAFONTu v knize METAFONTbook.
b) Nepamatujeme si přesně syntaxi svých vlastních definic, které při kreslení obrázků používáme, a také znovu a znovu zapomínáme pracně nabyté zkušenosti při kreslení obrázků.
c) Nepamatujeme si přesně názvy řídicích posloupností (a jejich parametry), pomocí nichž začleňujeme obrázky vytvořené v METAFONTu (a nejen v něm) do textu v TEXu.
d) Vše, co nám z výše uvedeného činí problémy, chceme mít někde (podle našeho názoru) přehledně sepsané. Důvod, proč toto uveřejňujeme v TEXbulletinu, je jednoduchý. Kde najít TEXbulletin víme. Kam jsme dali naše poznámky, to si nepamatujeme. Pokud naše tabulky nejběžnějších příkazů a zkušenosti shledá užitečným i někdo jiný, budeme rádi. Ještě je potřeba se zmínit o tom, že nejběžnějšími příkazy jsou pro nás ty příkazy, které jsme dosud použili. Seznam příkazů uvedeme v tabulkách na konci článku.

Celkový postup návrhu obrázku, vytváření obrázku a začlenění obrázku do textu budeme popisovat v tomto pořadí. Movitějším doporučujeme koupit si čtverečkovaný sešit, ve kterém si mohou své ručně kreslené návrhy obrázků schovávat, neboť po čase člověk zjistí, že ve věčném koloběhu života kreslí znovu a znovu pár obrázků. Nebo že potřebuje opakovat trik, který nedávno použil, a už si ho nepamatuje.

V každém případě doporučujeme nejdříve alespoň nahlédnout do výše citovaného článku Karla Horáka, neboť v něm je stručně popsán postup, jak s METAFONTem začít (jinak by tento článek mohl být spíše návodem, jak s METAFONTem skončit).

Podmínky, které nás obklopují

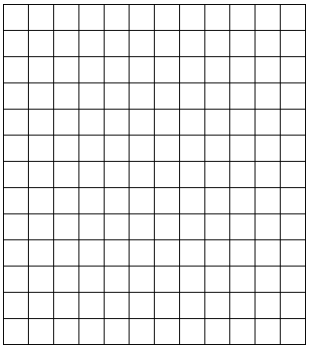
V naší praxi se vyskytují v převážné většině náčrtky, které si autor připraví jen tak od ruky na kusu papíru, a naší úlohou je tyto obrázky začlenit do matematického článku. Mnohdy se stane, že při korekturách autor řekne, že obrázek se mu zdá příliš malý, nebo příliš velký, nebo že by mohl mít jiný poměr stran. Jindy se stane, že by bylo potřeba obrázek mírně zmenšit, eventuálně zvětšit tak, aby hezky zapadal do místa, kam ho chceme umístit. Zvlášť nepříjemné jsou v tomto směru stránkové zlomy, když obrázek vychází přes něj a chybí pár milimetrů, aby se vešel tam, kde ho chceme mít.

Z tohoto vyplývá, že nejde ani o grafické zpracování výsledků, ani o kreslení složitých technických výkresů nebo o práci s přesnými rozměry.

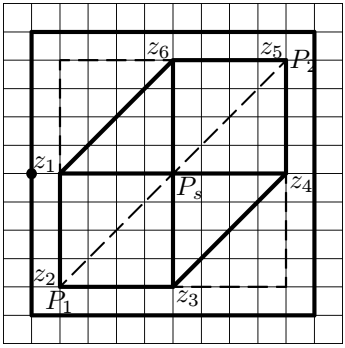
Je přirozené, že si autoři vymýšlejí různé popisky, které chtějí umístit do obrázku, a popřípadě pod obrázek napsat, co daný obrázek znázorňuje (pokud to z něj není patrné — což někdy nebývá). My bychom toto všechno chtěli dělat s co nejmenší námahou a co nejmenším úsilím. Proto jsme navrhli dva soubory maker, a to `incpic.mf` pro METAFONT a `incpic.tex` pro TEX. Tato makra jsme navrhovali s ohledem na jejich případnou vzájemnou spolupráci. Snad jen na okraj poznamenejme, že všechny obrázky v tomto článku jsou vytvořeny a začleněny do textu s pomocí výše zmíněných dvou maker. Je velice jednoduché předat spolu s tímto článkem jedno makro pro TEX, jedno makro a zdrojový soubor s obrázky pro METAFONT a nestarat se více o to, na jaké tiskárně bude TEXbulletin vytištěn. Tímto postupem nemá ani redakce téměř žádnou práci s tímto příspěvkem. Jak jsme tedy „vyráběli“ obrázky?

Jako relativně pohodlný se nám zdál tento postup, který budeme průběžně znázorňovat:

• Vezměme kus čtverečkovaného papíru (takový, jaký je zobrazen na obrázku 1, a nakresleme na něj od ruky obrázek, o který nám jde. Do tohoto obrázku zanesme také všechen popis, který tam má být (např. popis souřadných os, atd.). Jednoduchý náčrtek s popisem jsme se snažili



Obr. 1 Čistý papír na začátku.



Obr. 2 Připravený náčrtek.

znázornit na obrázku 2. V následujícím textu budeme uvádět v závorkách hodnoty platné pro tento obrázek.

• Zarámujeme obrázek i s popisem do rámečku a určíme, kde bude mít tento rámeček referenční bod (tj. najdeme nějaký bod na **levém** okraji rámečku, od kterého budeme pohodlně počítat čtverečky). V obrázku 2 je tento bod vyznačen malým puntíkem. Čtverečkovaná síť totiž bude představovat jednoduchý pomocný souřadný systém. Spočítáme vodorovně počet čtverečků v rámečku (v případě obrázku 2 je to 10) a svisle počet čtverečků od referenčního bodu k hornímu okraji rámečku (5).

• Uděláme si první odhad, jak veliký by měl obrázek ve skutečnosti být. (Řekněme 45×45 mm, přičemž část nad referenčním bodem bude vysoká 22,5 mm a pod referenčním bodem hluboká také 22,5 mm). Tyto rozměry nemusí mít nic společného s velikostí náčrtku. Tyto hodnoty budeme spolu s hodnotami z předchozího odstavce zapisovat do zdrojového textu pro METAFONT.

• Vyznačíme si v obrázku pomocné body, které budeme používat pro nakreslení křivek a čar obrázku. Tyto body je možné číslovat např. pořadovými čísly (na obrázku 2 to jsou body z_i). Uděláme (kafe a) rozvahu o tom, jakým postupem obrázek nejlépe nakreslit (všímáme si různých symetrií, stejných částí lišících se pouze otočením či posunutím, atd.) Dobře provedená úvaha na začátku ušetří později trochu práce. Pokud např. kreslíme více podobných obrázků, kdy jeden lze odvodit z druhého a lze postupovat od jednoduššího ke složitějšímu,

lze k zrychlení práce zadávání obrázků použít funkci kopírování bloků v editoru nebo aparátu definic METAFONTu.

• Napíšeme zdrojový soubor pro náš obrázek (obrázky) pro METAFONT (např. `clanek.mf` — příklad takového souboru bude uveden dále) a ladíme a ladíme a ladíme... Většinou je možné mít v jednom souboru více obrázků. Postupně provádíme dva kroky. Provedeme změnu vstupního souboru a přeložíme ho METAFONTem v režimu zobrazování na obrazovce, tj. příkazem

```
mf clanek
```

Když jsme s obrázkem spokojeni (doporučujeme ladit obrázky postupně jeden za druhým), vytvoříme font např. příkazem

```
mf &plain \mode=hplaser; mag=1.; input clanek
```

(konkrétní tvar záleží na výstupním zařízení, které používáme pro zobrazování a pro tisk) a do zvláštního textového souboru (například `clanek.obr`) napíšeme zdrojový text pro TEX, pro vysázení všech obrázků i s jejich případným popisem v obrázku a pod obrázkem (s využitím definic z makra `incpic.tex`). Tak, jak se ve fontu hromadí obrázky, hromadí se v tomto textovém souboru zápisy pro jejich vysázení. Tento soubor v TEXu odladíme a zkontrolujeme. Jak bude později patrné z popisu makra `incpic.tex`, měly by být tyto obrázky v pomocném souboru v pořadí, v jakém se budou začleňovat do textu. Příklad tohoto souboru bude také uveden dále.

• Když jsme hotovi s obrázky, napíšeme text dokumentu (například `clanek.tex`) a začleníme do něj obrázky jednoduchými příkazy definovanými v souboru `incpic.tex`. (Pokud není obrázků „příliš mnoho“ (jsme omezeni pamětí počítače), je možné načíst soubor s obrázky jednoduše pomocí příkazu `\input clanek.obr`.)

• Text odladíme. Podle potřeby tu a tam můžeme upravit v METAFONTovém zdrojovém textu rozměry obrázků tak, aby se hodily do textu. Nic víc není potřeba měnit — pouze po změnách METAFONTového zdrojového textu generovat znovu font s obrázky.

Co obsahuje METAFONTové makro `incpic.mf`

Snaha po maximálním zjednodušení nás vedla k zavedení nových definic a k vlastnímu nastavení některých parametrů. Protože se pokusíme odhalit čtenáři i zákulisí tohoto makra, může se stát, že pro někoho může být některá věta nesrozumitelná. Pokud se vám to stane, nemusíte hned sahat po METAFONTbooku nebo odkládat TEXbulletin, ale jednoduše takovou větu můžete přeskočit. Pravděpodobně pro vás není důležitá.

Činnost makra je následující:

• Definuje pracovní okno tak, aby obrazovka VGA (640 × 480 bodů) zobrazovala plochu asi 15 × 12 cm.

• Definuje režim práce `ouproof`, což je analogie režimu `proofmode` (tj. vytvářené znaky se zobrazují na monitoru, font se negeneruje). V režimu `ouproof` je nastaveno `showbox:=1; showmesh:=1;`, což má za následek (pokud později není nastaveno jinak), že vytvářené znaky budou zarámovány a budou obsahovat čtverečkovanou síť.

• Pokud není explicitně zadán režim práce (např. na příkazové řádce při spouštění METAFONTu), je zvolen režim práce `ouproof` (viz předchozí bod).

• Dále makro definuje nové příkazy, které jsou uvedeny níž.

• V závěru pak toto makro nastaví `font_size 100mm#;`, což znamená, že základní velikost fontu je 100 mm (tento údaj je důležitý, pokud chceme v TEXu zavádět tento font v nějaké explicitně uvedené velikosti).

• Je iniciováno makro `mode_setup;` nastavuje další parametry.

• Je nastavena hodnota `thinestline:=0.4pt;`, což je průměr pera pro kreslení nejtenčích čar.

Nyní následuje přehled dalších uživatelských definic a proměnných souboru `incpic.mf`, jejichž použití bude patrné také z příkladů uvedených dále:

• `beginch(`⟨*kód*⟩`,`⟨*šířka*⟩`,`⟨*výška*⟩`,`⟨*hloubka*⟩`,`⟨*x_čtv*⟩`,`⟨*y_čtv*⟩`);` uvozuje písmeno, které budeme vytvářet, přičemž:

⟨*kód*⟩ je kód znaku, který budeme vytvářet (např. `"A"`), je povoleno používat pouze prvních 20 znaků abecedy, tedy písmena A–T, tj. jeden font může obsahovat nejvýše 20 obrázků;

⟨*šířka*⟩ je rozměr, jak má být znak ve skutečnosti široký (např. `45mm#` — tento údaj musí být v „ostrých“ jednotkách, tj. jednotka míry musí být následována znakem `#`);

⟨*výška*⟩ je rozměr, jak má být znak ve skutečnosti vysoký (např. `22.5mm#` — opět v „ostrých“ jednotkách);

⟨*hloubka*⟩ je rozměr, jak má být znak ve skutečnosti hluboký (např. `22.5mm#` — opět v „ostrých“ jednotkách);

⟨*x_čtv*⟩ je vodorovný počet čtverečků v pomocném rámečku (např. `10`);

⟨*y_čtv*⟩ je svislý počet čtverečků v pomocném rámečku mezi referenčním bodem a horním okrajem rámečku (např. `5`);

Toto makro také zajistí to, že se nastaví hodnota proměnné `current transform` tak, že souřadnice všech zadávaných bodů mohou být zadávány v bezrozměrných souřadnicích (tj. čtverečkovaný papír je pro nás souřadný systém). Z toho plyne, že pokud nevíme, co děláme, raději s proměnnou `currenttransform` nepracujme. Dále toto makro uloží do „nějakých“ proměnných `fontdimen` absolutní rozměr čtverečku sítě pro pozdější využití TEXem. Toto opět nemusí čtenáře příliš zajímat, protože s těmito hodnotami nebude nikdy přímo pracovat (využívá je pouze makro v TEXu).

• `endch;` ukončuje písmeno. Pokud je hodnota proměnné `showmesh` pozitivní, zakreslí se do znaku celá síť odpovídající čtverečkovanému papíru (hustota této sítě je dána parametry ⟨*x_čtv*⟩ a ⟨*y_čtv*⟩ v makru `beginch`). Pokud je hodnota proměnné `showbox` pozitivní, nakreslí kolem znaku rámeček.
`beginch` a `endch` znamenají také začátek a konec skupiny, tj. nastavení proměnných mezi nimi neovlivní nastavení proměnných v dalších znacích.

• `axes`⟨*bod*⟩ nakreslí souřadné osy, které se budou protínat v bodě ⟨*bod*⟩. Toto makro bylo v podstatě převzato z článku K. Horáka, ale po mírné úpravě vyplývající z koncepce makra `incpic.mf`.

• `arrow(`⟨*délka*⟩`,`⟨*úhel*⟩`)` je cesta pro šipku s délkou ⟨*délka*⟩ a směrem daným číslem ⟨*úhel*⟩. Co je to cesta, bude vysvětleno později. Toto je po úpravě také převzato z výše zmiňovaného článku.

• `sharparrow` je cesta ve tvaru šipky. Tato šipka je tvarově jiná, než `arrow`. Tato šipka je standardně dlouhá 3mm a ve své základní poloze směřuje vpravo (hrot je její počátek).

• `vector(`⟨*bod_1*⟩`,`⟨*bod_2*⟩`);` nakreslí vektor (tj. úsečku s šipkou na konci) začínající v bodě ⟨*bod_1*⟩ a končící v bodě ⟨*bod_2*⟩.

• `cerchovane(`⟨*bod_1*⟩`,`⟨*bod_2*⟩`);` spojí body ⟨*bod_1*⟩, ⟨*bod_2*⟩ čerchovanou **úsečkou**.

• `dashed(`⟨*bod_1*⟩`,`⟨*bod_2*⟩`);` spojí body ⟨*bod_1*⟩, ⟨*bod_2*⟩ čárkovanou **úsečkou**.

• `penfordots;` zvolí kruhové pero o průměru `10thinestline`. Jak již název napovídá, používáme toto pero na kreslení teček.

• `thickpen;` zvolí kruhové pero o průměru `4thinestline`. Toto pero používáme na kreslení tlustých čar.

• `middlepen;` zvolí kruhové pero o průměru `2thinestline`. Toto pero používáme na kreslení středně tlustých čar.

• `thinpen;` zvolí kruhové pero o průměru `thinestline`. Toto pero používáme na kreslení tenkých čar (popřípadě na kreslení čtverečkované sítě a rámečku kolem znaku).

• `drawdots(`⟨*bod_1*⟩`,`⟨*bod_2*⟩`,...);` kreslí postupně v bodech ⟨*bod_1*⟩, ⟨*bod_2*⟩, ... otisk zvoleného pera.

• `cdrawdot(`⟨*bod_1*⟩`);` nakreslí v bodě ⟨*bod_1*⟩ otisk obrysu zvoleného pera (tj. pro kruhové pero udělá v daném bodě kroužek, nikoliv tečku).

• `cdrawdots(`⟨*bod_1*⟩`,`⟨*bod_2*⟩`,...);` bude kreslit postupně v bodech ⟨*bod_1*⟩, ⟨*bod_2*⟩, ... otisky obrysu zvoleného pera.

Nejběžnější příkazy pro kreslení obrázků

V METAFONTu lze používat proměnné pro různé typy objektů. Seznam možných deklarací je uveden v tabulce 5.

Abychom mohli vůbec něco kreslit, je nutno zadávat body v obrázku. Body zadáváme buď explicitně (např. `(4.5,-12)`) nebo je označujeme písmenem `z` a číslem (něco jako index, ale indexem mohou být i desetinná čísla), např. `z5, z1.4, z[k]` (v tomto posledním případě si zjednodušíme psaní a v textu budeme takový bod označovat jako `z`*k*). Každý bod `z`*k* má dvě souřadnice (`x`*k*`,y`*k*), a tedy výše označené body mají ekvivalentní zápis `(x5,y5), (x1.4,y1.4), (x[k],y[k])`. Body je možné označovat i jinými písmeny než `z`, ale v takovém případě je možné získat jejich jednotlivé souřadnice pomocí funkcí `xpart` a `ypart` uvedených v tabulce 7.

V METAFONTu existuje jednak znak pro přiřazení hodnoty proměnné (`:=`) a jednak znak pro rovnost dvou veličin (`=`). Jednotlivé souřadnice se mohou vyskytovat na obou stranách přiřazení i rovností, body (tj. dvojice souřadnic) pouze v rovnostech. Na základě přiřazení a rovností si METAFONT dopočítává souřadnice bodů. Obecně platí pravidlo, že rovnic a přiřazení musí být tolik, aby jednoznačně určovaly všechny body, které používáme. METAFONT umí řešit soustavy lineárních rovnic, k jejichž zápisu je možné využít právě znaku `=`. Na bod lze pohlížet také jako na vektor, neboť spojnice bodu s počátkem určuje směr i velikost. Názvy pro některé body (vektory), které se často používají, jsou uvedeny v tabulce 2.

Prostředky pro zadávání bodů mohou být velmi rozmanité. U bodů, u kterých je jejich poloha zcela zřejmá a lze je zapsat snadno pomocí souřadnic, můžeme tak učinit. Pro body, u kterých je určení jejich souřadnic obtížné, ale které splňují určité vlastnosti (např. leží v průsečíku přímek či křivek, atd.) stačí popsat tyto vlastnosti a nechat určení souřadnic bodů METAFONTu. S body je možné zacházet jako s vektory. S body lze provádět základní operace (sčítat, odčítat, násobit číslem, ...).

S proměnnými je možné provádět různé operace a úkony, seznam nejběžnějších operací je v tabulce 7.

Pod pojmem **cesta** budeme rozumět spojnici dvou nebo více bodů. Spojnice mezi dvěma body může být úsečka (což naznačujeme znaky `--`) nebo křivka, která je založena na Bezièrových polynomech třetího stupně (toto naznačujeme znaky `..`). Pokud spojujeme více bodů křivkou, bývá tato křivka ve vnitřních bodech hladká (tj. nemá v nich zlomy — viz dál). Pokud spojujeme dva body křivkou, můžeme průběh křivky mezi těmito dvěma body ovlivnit. Můžeme předepsat směr, v jakém má křivka do daného bodu vstupovat, resp. z něj vycházet, nebo můžeme předepsat „napětí“ mezi dvěma body v křivce, které říká, jak je křivka deformovaná. Protože pro nakreslení křivky spojující dva body si METAFONT vypočítá dva pomocné body (které nemusí ležet na křivce), je možné ovlivnit průběh křivky i předepsáním těchto pomocných bodů. Jednotlivé elementy, z nichž se může skládat popis cesty je v tabulce 4. Příklady cest budou uvedeny níž. Popsat cestu znamená říct:

Z bodu vychází (popř. v předepsaném směru) křivka, která může mít předepsané napětí mezi body (nebo se řídit podle předepsaných pomocných bodů) a přichází (popřípadě s předepsaným směrem) do dalšího bodu, a z něj vychází (popř. v předepsaném směru) křivka

... a přichází (popřípadě s předepsaným směrem) do posledního bodu. Pokud je místo posledního bodu napsáno `cycle`*, znamená to, že je posledním bodem výchozí bod, a v takovém případě (pokud není předepsán směr) je uzavření cesty hladké.*

Pokud potřebujeme opakovat některou část vícekrát (např. zadání stejných bodů v podobných obrázcích), je možné využít aparát definic METAFONTu. Zde uvedeme pouze velice jednoduchý příklad definice bez parametrů. Definice začíná slovem `def`, za kterým následuje název definice, za kterým je rovnítko. Tělem definice je vše za tímto rovnítkem až po uzavírající slovo `enddef;`. Jestliže napíšeme například

```
def body =     % definice bodů společných pro více obrázků
    z1=(2,0); z2=(2,-4); z3=(6,-4);  z4=(10,0); z5=(10,4);
    z6=(6,4); z7=(6,0);  z8=(10,-4); z9=(2,4);
  enddef;
```

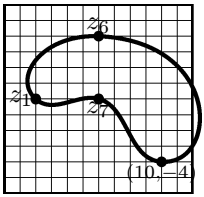
pak slovo `body;` uvedené kdekoliv je totéž, jako bychom v takovém místě napsali tělo definice, tj. přiřazení souřadnic bodům `z1`, ..., `z9`.

Nyní jsme již řekli dost informací na to, abychom mohli dát snadnou odpověď na čtenářovu zvědavou otázku, jak jsme nakreslili obrázky 1 a 2. Příkaz `draw` kreslí zadanou křivku zvoleným perem, příkaz `drawdot` udělá „otisk“ zvoleného pera v daném bodě — příkazy pro kreslení a mazání jsou uvedeny v tabulce 3. Zde je zdrojový text souboru se dvěma obrázky (jde o obrázky 1 a 2):

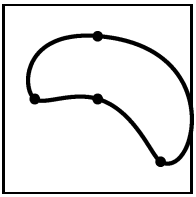
```
\input incpic

def body =     % definice bodů společných pro více obrázků
    z1=(2,0); z2=(2,-4); z3=(6,-4);  z4=(10,0); z5=(10,4);
    z6=(6,4); z7=(6,0);  z8=(10,-4); z9=(2,4);
  enddef;

beginch("A",40mm#,45mm#,0mm#,12,13);
showmesh:=1;            % zobrazit čtverečkovanou síť
endch;

beginch("B",45mm#,22.5mm#,22.5mm#,12,6);
showmesh:=1;            % zobrazit čtverečkovanou síť
```

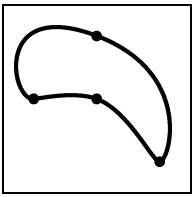
```
body;                    % definice bodů
thickpen;                % volba tlustého pera
draw (1,-5)--(11,-5)     % rámeček okolo obrázku
      --(11,5)--(1,5)--cycle;
draw z1--z2--z3--z4--z5--z6--cycle; % šestiúhelník
draw z1--z4; draw z3--z6;           % kříž v šestiúhelníku
middlepen;               % volba středně tlustého pera
dashed(z3,(10,-4));      % čárkované čáry
dashed((10,-4),z4);  dashed(z6,(2,4));
dashed((2,4),z1);    dashed(z2,z5);
penfordots; drawdot (1,0);  % znázornění referenčního bodu
endch;
end
```

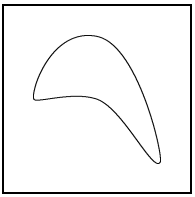
Následující série obrázků 3–6 ukazuje, jak jsme vytvářeli obrázek profilu lopatky turbíny (naše zkušenost je, že v tomto případě metoda po-



Obr. 3 Poprvé.


Obr. 4 Podruhé.


Obr. 5 Potřetí

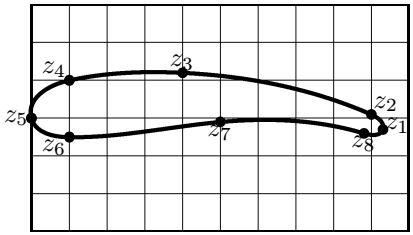

Obr. 6 A je to.

Obr. 3
```
draw z1..z7..(10,-4)..z6..cycle;
```
Obr. 4
```
draw z1..tension 2 and .8..z7..
        tension 8 and 3 ..(10,-4)..tension 2 and 1 ..z6..
        tension 8 and 2..cycle;
```
Obr. 5
```
draw z1..tension 5 and .9..z7..
        tension.9 and 5 ..(10,-4)..tension 5 and 1 ..z6..
        tension .8 and 2..cycle;
```
Obr. 6
```
draw z1..tension 5 and .9..z7
        ..tension .9 and 5 ..(10,-4)..tension 9 and 1.5
        ..z6..tension 1.2 and 5 ..cycle;
```

kusu a omylu vede nejrychleji k cíli a k nasbírání potřebných zkušeností — pak je těch omylů méně). Pod obrázky jsou uvedeny příkazy, kterými

byly dané křivky nakresleny. Body `z1, z6, z7` jsou stejné jako ve zdrojovém textu pro písmeno `"B"` výš. Na posledním obrázku je to, k čemu jsme chtěli dospět (tenkou čáru jsme zvolili proto, aby bylo vidět, že nám šlo o hladkou křivku). Zároveň poznamenejme to, že ze všech pokusů, které jsme dělali dříve s různými typy popisu průběhu křivek (zadávání směrů, více bodů, atp.) nás nejrychleji dovedlo k cíli používání příkazu `tension` (vytvoření všech čtyř obrázků trvalo u všech všudy 5 minut).

Protože už víme, co je to cesta, je vhodné se ještě zmínit o možnosti popisu jednotlivých bodů na cestě. Jestliže nějaká cesta je určena k body, pak existuje spojité a vzájemně jednoznačné přiřazení čísel z intervalu $\langle 0, k-1\rangle$ bodům této cesty, přičemž platí, že celá čísla intervalu se zobrazují na zadané body cesty. Budeme říkat, že každému času $t \in \langle 0, k-1\rangle$ odpovídá bod na cestě. Na obrázku 7 je profil křídla, který lze parametrizovat na intervalu $\langle 0, 8\rangle$ (neboť je zadáno 8 bodů a jde o uzavřenou cestu, tj. je spojeno 9 bodů — viz níž).



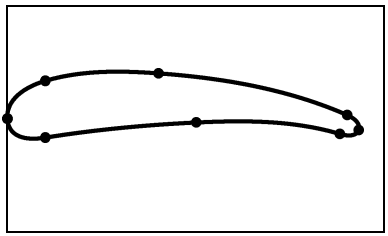
Obr. 7 Profil křídla.

Použití některých operací z tabulek 8 a 7 uvedeme na příkladech. To, k čemu směřujeme, je na obrázku 9. Povšimli jsme si, že všechny tři profily si jsou velmi podobné a liší se natočením, umístěním a měřítkem. Proto jsme se začali zabývat nejdříve náčrtkem jednoho profilu, který je vidět na obrázku 7. Pro lepší názornost ukazujeme i čtverečkovanou síť. Zdrojový text tohoto obrázku pro METAFONT má tvar:

```
def proprofil = path profila;
    z1=(9.3,-0.3); z2=(9.0,0.1); z3=(4.0,1.2); z4=(1,1);
    z8=(8.8,-0.4); z6=(1,-0.5);  z7=(5,-0.1);  z5=(0,0);
enddef;

beginch("G",50mm#,15mm#,15mm#,10,3);
showmesh:=1;
proprofil;
profila:=z1..z2..z3..z4..z5..z6..z7..z8..cycle;
thickpen;  draw z1..z2..z3..z4..z5..z6..z7..z8..cycle;
penfordots; drawdots (z1,z2,z3,z4,z5,z6,z7,z8);
```

```
endch;
```

Poznamenejme jen, že druhý řádek způsobil, že je v obrázku viditelná i síť. Proměnnou `profila`, definovanou jako cesta, jsme zavedli právě kvůli tomu, že různými deformacemi a posunutími této cesty dosáhneme všech tří profilů (proto nám ani nevadí, že je tento základní profil („nos“ profilu) umístěn v počátku).



Obr. 8 Upravený profil.

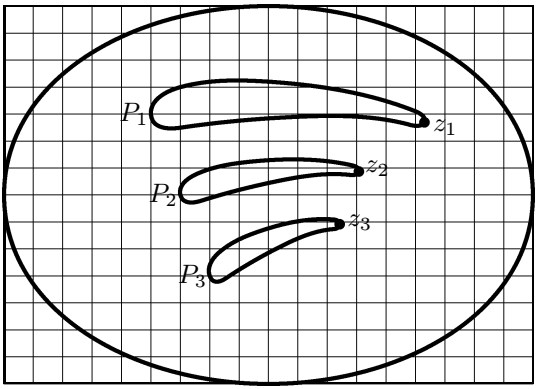
Protože se nám tento profil ještě nelíbil, mírně jsme upravili definici cesty takto:

```
profila:=z1..z2..z3..z4..z5..
            tension 0.88..z6..z7..z8..cycle;
```

Na obrázku 8 je vidět malá změna, kterou pozměněním zdrojového textu dosáhli. Protože se nám tento výsledek líbil, přistoupili jsme k nakreslení celého obrázku 9. Zdrojový text je uveden pod ním.

Jak je vidět srovnáním se zdrojovým textem předcházejícího znaku, změny jsou nepatrné (tj. všechny tři profily byly nakresleny po natočení, zmenšení a posunutí základního profilu odladěného předem). Zde nastává malá obtíž, protože ve výsledném obrázku chceme mít popsány body, které jsme explicitně nezadali. Jejich umístění na čtverečkovaném papíře v náčrtku a po vygenerování se neshodují, protože vygenerovaný obrázek není přesnou kopií náčrtku od ruky. Vyřešení tohoto problému je snadné, neboť si můžeme nechat při generování znaku zobrazit také čtverečkovanou síť (toto stačí udělat na obrazovce) a poznamenat si přibližné souřadnice bodů, do kterých umístíme popis. Toto byl jeden z hlavních důvodů, proč jsme zaváděli kreslení čtverečkované sítě. Výhodou tohoto postupu je to, že neustále pracujeme s relativními jednotkami (čtverečky), které jsou nezávislé na skutečných rozměrech obrázku. Proto změna rozměrů obrázku neovlivní umístění popisu a žádné další změny nebude potřeba v dokumentu ani v budoucnu provádět (tj. ani po změně skutečných rozměrů obrázku). Toto ovšem není jediný způsob, ale nám se zdál nejjednodušší (vzhledem k četnosti výskytu tohoto případu popisu).

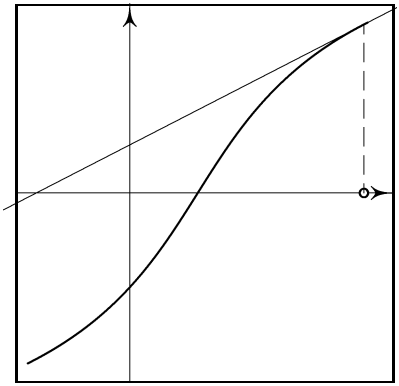
```
beginch("I",70mm#,25mm#,25mm#,18,7);
```



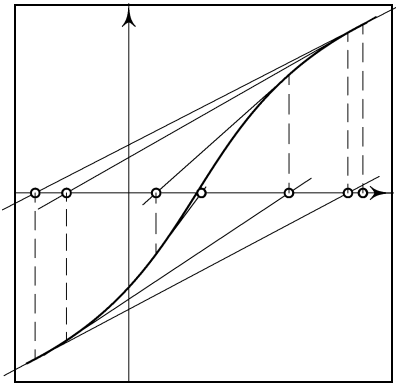
Obr. 9 Posunuté profily.

```
showmesh:=1;    proprofil;
profila:=z1..z2..z3..z4..z5..tension
                        0.88..z6..z7..z8..cycle;
z9=(5,3);     z10=(6,0);    z11=(7,-3);
thickpen;     draw profila shifted z9;
draw profila rotated 10 scaled 0.66 shifted z10;
draw profila rotated 20 xscaled 0.5 yscaled 0.66
                                        shifted z11;
penfordots;   drawdot z1 shifted z9;
drawdot z1 rotated 10 scaled 0.66 shifted z10;
drawdot z1 rotated 20 xscaled 0.5 yscaled 0.66
                                        shifted z11;
thickpen;
draw (0,0)..(0.5w,-d)..(w,0)..(0.5w,h)..cycle;
endch;
```

V předposlední ukázce je použito několik užitečných funkcí z tabulky 6 a potvrzují se naše slova ze začátku. Přesto, že následující dva obrázky vypadají dost odlišně, zdrojový text pro METAFONT se podstatně liší pouze ve dvou řádcích. První obrázek znázorňuje jeden krok Newtonovy iterační metody na hledání nulových bodů funkce jedné proměnné. Po letmém nahlédnutí do tabulek z konce článku bude zřejmě příslušný zdrojový text zcela čitelný. Pozoruhodné je např. to, že křivka je zadána jen dvěma body (čím méně, tím lépe). Zadáním směrů v krajních bodech



Obr. 10 Jeden krok Newtonovy iterační metody.



Obr. 11 Newtonova iterační metoda (druhého řádu).

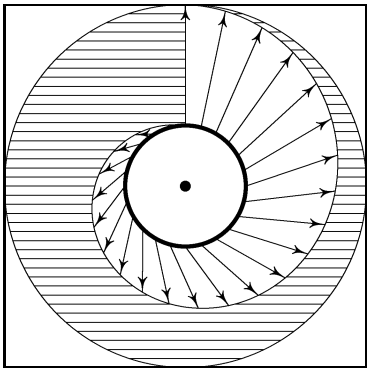
křivky jsme zaručili, že proces zobrazený na druhém obrázku bude konvergentní (čtenáře, jimž se zdá, že používáme mnoho matematických pojmů, prosím za prominutí). Dále je ve zdrojovém textu vidět, jak konstruujeme tečnu ke křivce.

```
beginch("J",60mm#,30mm#,30mm#,10,10);
path krivka;
numeric a,b;
thinpen;    axes (3,0);              % kreslení os
z1=(0.3,-9); z2=(9.3,9);             % krajní body křivky
krivka:=z1{(x2,0)-z1}..{z2-(x1,0)}z2;  % křivka
middlepen; draw krivka;              % vykreslení křivky
z3=(9.2,0);                          % první bod na ose
  penfordots; cdrawdot z3;   % a jeho vykreslení
  save a,b; (a,b) = krivka  % určení časů průsečíku křivky
      intersectiontimes ((x3,-d)--(x3,h));  %   a svislice
  z13 = point a of krivka;  % určení průsečíku na křivce
  thinpen;  dashed(z3,z13);          % čárkovaná svislá čára
  z23 = direction a of krivka;       % směr tečny ke křivce
  z4 = z13+whatever*z23;
  z4 = whatever*z3;                  % průsečík tečny s osou
  draw 1.1[z13,z4]--1.1[z4,z13];     % kreslení tečny
endch;
```

Nyní je celkem zřejmé, že obrázek 11 dostaneme, pokud nahradíme číslo `3` indexem `[i]`, číslo `4` indexem `[i+1]`, číslo `13` indexem `[i+10]`, číslo `23` indexem `[i+20]`, a necháme `i` probíhat celá čísla od `3` do `9`. Tedy ve zdrojovém textu pro obrázek 11 budou řádky, které jsou mírně odsazené od levého okraje, nahrazeny těmito řádky:

```
for i=3 upto 9 :       % počet tečen
  penfordots; cdrawdot z[i];
  save a,b; (a,b) = krivka intersectiontimes
                              ((x[i],-d)--(x[i],h));
  z[i+10]= point a˜of krivka;
  thinpen;  dashed(z[i],z[i+10]);
  z[i+20] = direction a˜of krivka;
  z[i+1]= z[i+10]+whatever*z[i+20]; z[i+1]=whatever*z[i];
  draw 1.1[z[i+10],z[i+1]]--1.1[z[i+1],z[i+10]];
endfor;
```

Na začátku jsme si řekli, že rámeček kolem náčrtku uděláme tak, aby obsahoval i případný popis. Jedna z praktických námitek byla, že je o něco pohodlnější udělat rámeček pouze kolem samotného obrázku. Protože existují dobré důvody pro výše popsaný postup, ukážeme na posledním příkladě způsob, jak vyhovět tomuto požadavku a mít počátek čtverečkovaných souřadnic kdekoliv v obrázku.



Obr. 12 Spirála.

Uvažujme velmi jednoduchý náčrtek, který je na obrázku 12. Referenční bod je znázorněn velkou tečkou a předpokládejme, že síť (kterou jsme z důvodů přehlednosti do obrázku nemalovali) je velikosti 6×6 čtverečků.

```
beginch("L",48mm#,48mm#,0mm#,6,6);
currenttransform:=  % posunutí souřadného systému
      currenttransform shifted (3aux_sx,3aux_sy);
thinpen;            % pero pro vodorovné čáry
```

```
n:=20;              % počet vodorovných čar nad osou
for i=-n upto n:    % cyklus pro kreslení vodorovných čar
  draw (-3,i*3/n)--(3,i*3/n);  %       od okraje k okraji
endfor;
erase filldraw      % vymazání "rohů čtverce"
   (0,-3)--(3,-3)--(3,3)--(-3,3)--(-3,-3)--(0,-3)
         ..(-3,0)..(0,3)..(3,0)..cycle;
path spirala,uplnaspirala;
numeric a,b;
r:=1; b:=2;         % poloměr vnitří a vnější kružnice
uplnaspirala:=(0,r+b)  % spirála a uzavírající svislice
   ..(r+.75b,0)..(0,-r-.5b)..(-r-.25b,0)..(0,r)--cycle;
spirala:=subpath (0,4) of uplnaspirala;  % spirála
erase fill uplnaspirala;  % vymazání vnitřku spirály
thinpen  draw fullcircle scaled(2r+2b);% vnější kružnice
thickpen draw fullcircle scaled 2r;    % vnitřní kružnive
thinpen  draw spirala;                 % spirála
n:=24;                      % počet vektorů
ar:=360/n;                  % přírustek ve stupních
for i=0 upto n-1:           % pro vykreslení vektorů
  a:=90-i*ar;               % úhel a příslušný počáteční
  z[3i]=r*(cosd a,sind a); %   bod na vnitřní kružnici
  z[3i+1]=                % "koncový" bod na vnější kružnici
     (r+2)*(cosd(a+50i/n),sind(a+50i/n));
  z[3i+2]=                % skutečný koncový bod na spirále
     (z[3i]--z[3i+1]) intersectionpoint spirala;
  vector(z[3i],z[3i+2]);   % vykreslení vektoru
endfor;
penfordots drawdot (0,0); % počátek souřadného systému
endch;
```

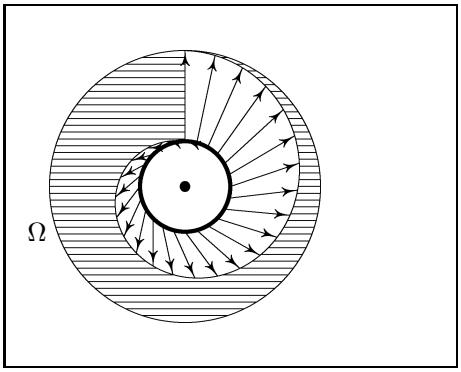
V druhém a třetím řádku je definováno posunutí všeho, co se bude dále kreslit (a tedy i počátku). Aniž bychom se pouštěli do hlubšího vysvětlování, zdůrazněme, že v tomto příkazu je nutno uvádět ve vektoru posunutí tajuplné jednotky `aux_sx`, `aux_sy` přesto, že čísla udávají počet čtverečků. Jestliže změníme rozměr obrázku a počty čtverečků v svislém i vodorovném směru a změníme posunutí počátku, tj. změníme první tři řádky na

```
beginch("M",60mm#,48mm#,0mm#,10,8);
currenttransform:=
            currenttransform shifted (4aux_sx,4aux_sy);
```

Výsledek pak bude vypadat tak, jak je nakresleno na obrázku 13.

Jak začlenit obrázky do textu

Protože začleňování obrázků do textu pomocí makra `incpic.tex` tvoří samostatnou kapitolu, uvedeme jej v příštím čísle TEXbulletinu. Zde pouze uvedeme nejjednodušší způsob, jak vysázet vytvořený obrázek. Pokud jsme vytvořili například v znaku `A` obrázek v souboru `clanek.mf`, pak po zpracování METAFONTem a programem `gftopk` získáme font `clanek.pk` a příslušnou metriku `clanek.tfm`. Potom na začátek souboru pro TEX stačí napsat `\font\obr=clanek` a v místě, kam chceme obrázek umístit, napsat `{\obr A}`.



Obr. 13 Upravená spirála.

Jak si makra opatřit

Makra `incpic.mf` a `incpic.tex` je možné získat v Matematickém ústavu UK.

Tabulky nejběžnějších příkazů

`+`	sčítání
`-`	odčítání
`*`	násobení
`/`	dělení
`**`	umocňování (např. `a**3` je totéž, co `a*a*a`)
`++`	pythagorejské sčítání (`a++b` znamená `a*a+b*b`)
`+-+`	pythagorejské odčítání (`a+-+b` znamená `a*a-b*b`)
λ`[a,b]`	lineární kombinace (`3[a,b]` znamená `a+3(b-a)`)
`abs a`	absolutní hodnota čísla `a` (tj. $\|a\|$)
`sqrt a`	je odmocnina z čísla a (tj. $\sqrt{a}$)
`sind 40`	je číslo $sin\,40°$
`cosd 40`	je číslo $cos\,40°$
`dir 40`	je vektor $(cos\,40°, sin\,40°)$

Tab. 1 Operace s čísly

`a[z1,z2]`	je bod `z1+a(z2-z1)` (`a` je číslo).
`whatever`	je proměnná, která při každém použití nabývá nějaké hodnoty (její hodnota je určena zpravidla ostatními vztahy). Např. rovnost `z1=z2+whatever*(z3-z4)` říká, že body `z1` a `z2` leží na přímce se směrnicí `z3-z4` (body `z3`, `z4` **musí** být už určené).
`origin`	je totéž, co bod `(0,0)`
`right`	je totéž, co bod `(1,0)`
`up`	je totéž, co bod `(0,1)`
`left`	je totéž, co bod `(-1,0)`
`down`	je totéž, co bod `(0,-1)`

Tab. 2 Předdefinované hodnoty některých směrů a hodnot.

`pickup pencircle scaled č`	volba kruhového pera o průměru č
`drawdot z;`	nakreslí tečku v bodě `z` zvoleným perem (obecně nemusí být kruh)
`draw p;`	nakreslí cestu `p` zvoleným perem (tj. střed pera se pohybuje po zadané cestě)
`fill p;`	vyplní uzavřenou cestu `p` (tj. uzavřená cesta vymezuje plochu, která se vyplní)
`filldraw p;`	je totéž jako `fill p; draw p;`
`erase draw p;`	vymaže plochu ohraničenou obrysem zvoleného pera pohybujícího se po cestě `p`
`erase fill p;`	vymaže plochu ohraničenou uzavřenou cestou `p`

Tab. 3 Příkazy pro kreslení

`z[k]`	bod o souřadnicích `(x[k],y[k])`
`cycle`	znamená uzavřít křivku (vrátit se do prvního bodu)
`--`	úsečka mezi body
`..`	křivka mezi body
`..{dir 30}`	křivka přichází do bodu se směrnicí 30°
`..{z6}`	křivka přichází do bodu se směrnicí danou vektorem `z6`
`{dir 30}..`	křivka vychází z bodu se směrnicí 30°
`{z6}..`	křivka vychází z bodu se směrnicí danou vektorem `z6`
`tension 1.6 and .8`	křivka má napětí `1.6` u levého bodu a napětí `0.8` u druhého bodu
`tension 2`	je totéž co `tension 2 and 2`
`control z3 and z4`	křivka se má řídit pomocnými body `z3` a `z4`
`fullcircle`	jednotková kružnice se středem v počátku

Tab. 4 Elementy cest.

Deklarace	Do proměnných `a` a `b` lze ukládat	Příklad
`numeric a,b;`	čísla	`a=4.8; b:=-12;`
`pair a,b;`	body	`a=(3.4,-6);`
`path a,b;`	cesty	`a:=z1..z2--z3;`
`boolean a,b;`	logické hodnoty	`b:=true;`
`pen a,b;`	tvar pera	`a:=pencircle scaled 1pt;`
`picture a,b;`	celé obrázky	
`string a,b;`	řetězce	
`transform a,b;`	transformace	

Tab. 5 Deklarace proměnných.

`reverse p`	znamená cestu `p` probíhanou od posledního bodu k prvnímu
`p1&p2`	výsledkem je cesta vzniklá spojením cest `p1` a `p2` (koncový bod první cesty musí být počátečním bodem druhé cesty).
`length p1`	je počet bodů, které cesta spojuje, zmenšený o 1
`subpath (a,b) of p`	je ta část cesty `p`, která odpovídá časům z intervalu (a, b).
`point t of p`	je bod na cestě, který odpovídá času t.
`direction t of p`	je vektor udávající směrnici tečny k cestě v bodě odpovídajícímu času t
`directiontime z of p`	je číslo (čas) t, pro který má tečna v odpovídajícím bodě cesty směr vektoru `z`
`p1 intersectiontimes p2`	je dvojice časů (s, t), kde čas s přísluší cestě `p1` a odpovídá průsečíku cest `p1` a `p2` a čas t přísluší cestě `p2` a odpovídá průsečíku cest `p1` a `p2`
`p1 intersectionpoint p2`	je bod (s, t), ve kterém se cesty `p1` a `p2` protínají

Tab. 6 Operace s cestami (zde písmeno `p, p1, p2` jsou nějaké cesty).

`+`	sčítání vektorů `z1+z2` je totéž, co `(x1+x2,y1+y2)`
`-`	odčítání vektorů `z1-z2` je totéž, co `(x1-x2,y1-y2)`
`*`	násobení vektoru skalárem `3*z1+2(1,4)` je totéž, co `(3x1+2,3y1+8)`
`abs z`	absolutní hodnota vektoru `z` (tj. $\|z\| = \sqrt{x^2+y^2}$)
`length z`	je totéž, co `abs z`
`angle z`	úhel, který svírá vektor `z` s kladnou poloosou x
`xpart z`	je x-ová souřadnice vektoru `z`
`ypart z`	je y-ová souřadnice vektoru `z`
`z1 dotprod z2`	je skalární součin vektorů `z1` a `z2` (tj. `x1*y1+x2*y2`)

Tab. 7 Operace s body (vektory). Transformace (operace v dolní části tabulky jsou elementárními operacemi, z nichž je možné skládat transformace).

`identity`	identická transformace
`scaled č`	násobení číslem `č` v obou složkách (roztažení či smrštění)
`xscaled č`	násobení číslem `č` v x-ové souřadnici
`yscaled č`	násobení číslem `č` v y-ové souřadnici
`shifted z`	posunutí o vektor `z`
`rotated č`	otočení o úhel č°.
`reflectedabout(z1,z2)`	znamená zrcadlový obraz vzhledem k přímce určené body `z1` a `z2`
`rotatedaround(z1,č)`	znamená otočení o úhel č° okolo bodu `z1`

Tab. 8 Elementární transformace

Oldřich Ulrych