České vysoké učení technické v Praze
Fakulta strojní

BAKALÁŘSKÁ PRÁCE

Petr Čížek

Parametrický CAD model radiální turbíny

CD příloha: komentované úryvky ze skriptu

Ústav technické matematiky

Vedoucí bakalářské práce: doc. Ing. Ivana Linkeová, Ph.D.

Studijní program: Teoretický základ strojního inženýrství

Praha 2017

Obsah

Úvod

Tento je stručný stručný komentář ke skriptu vytvořenému v rámci bakalářské práce. Skript je také součástí tohoto CD. Jsou vynechány všechny části, které se opakují (např. je ukázáno jen jednou, jak se ukládá bod do proměnné, jak se vytváří Beziérova křivka atd.) a také ty, které nemají velkou důležitost pro výsledný model (např. práce s vrstvami). Následuje úvod do principů RS použitých v této práci.

Důležité je poznamenat, že skript byl vytvořen v Rhinoceros 4 a pouze tam je plně zkoušen a funkční.

1. RhinoScript

RhinoScript[1] je zásuvný modul programu Rhinoceros, který využívá WindowsScript (resp. Visual Basic Script), aby volal a za sebe řetězil jednotlivé příkazy z Rhina. RS nerozlišuje malá a velká písmena. Tento text vychází z nápovědy pro RS [2].

1.1 Proměnné

1.1.1 Volání proměnných

Druhy proměnných v RS se nijak zvlášť neliší od proměnných používaných v jiných programovacích jazycích. Následuje přehled a důvod použití jednotlivých druhů proměnných v této práci. Názvy proměnných byly z důvodu přehlednosti upraveny tomu, o jaký druh proměnné se jedná (viz tabulka 1.1).

Tabulka 1.1: Vlastnosti původních ploch rotorových lopatek

druh proměnné	upravený název proměnné A
celé číslo	`intA`; `I_A`
reálné číslo	`realA`; `R_A`
textový řetězec	`strA`; `S_A`
pole	`arrA`; `A_A`
RGB barva	`ColorA`

Celá a reálná čísla - tyto druhy proměnných byly použity pro uchovávání hodnot čísel, jako jsou zadávané hodnoty parametrů modelu, mezivýsledky a hodnoty jednotlivých souřadnic bodů). Celé číslo 5 se do proměnné `intA` přiřadí následujícím způsobem.

```
intA = 5
```

Reálné číslo 5.5 se do proměnné `realA` zapíše jako

```
realA = 5.5 .
```

Pole reálných čísel - do těchto proměnných byly nejčastěji ukládány body, tedy pole třech čísel, přičemž nultý prvek[3] odpovídal hodnotě souřadnice x, první prvek souřadnici y a druhý prvek souřadnici z. Pole reálných čísel 1, 1.5, 2 se přiřadí do proměnné `arrA` následujícím způsobem.

```
arrA = Array (1, 1.5, 2)
```

Textový řetězec - tato proměnná má v RS značný význam. Ukládají s zde jména vrstev a především identifikátory objektů[4]. Lze tedy říci, že do těchto proměnných byly ukládány odkazy na všechny objekty, které byly programem

[1] dále jen RS

[2] RSHelp

[3] prvky polí jsou v RS očíslovány od nuly, n-tý prvek pole A se značí `A(n-1)`

[4] Každý objet vytvořený v Rhinu má svůj jednoznačný identifikátor, který je přiřazen při vzniku objektu a může vypadat například takto: C82573C6-C93C-4f16-A613-2017E90597EC. Kdykoli je poté potřeba na tento objekt v programu odkázat, je tak činěno pomocí identifikátoru.

vytvořeny. Textový řetězec "C82573C6"se do proměnné `strA` uloží následujícím způsobem.

```
strA = "C82573C6"
```

Pole textových řetězců - do těchto proměnných se většinou ukládá série odkazů na objekty, například identifikátory ploch vzniklých rozdělením („rozříznutím“, anglicky „split“) jedné plochy. Pole textových řetězců "9b9a2a", "9760"a "46b3"se přiřadí do proměnné `arrA` jako

```
arrA = Array ("9b9a2a", "9760", "46b3")
```

Pole polí - pokud je potřeba zadat jako argument příkazu vyšší množství bodů, tyto body (tedy tříprvková pole) se naskládají do pole. Představují-li proměnné x, y, z pole reálných čísel, tyto se uloží do proměnné `arrA` následujícím způsobem.

```
arrA = Array (x, y, z)
```

RGB barva - barva je ve výpočetní technice nejčastěji specifikována poměrem červené, zelené a modré (každá ze zmíněných barev je reprezentována parametrem, který může dosahovat od 0 do 255). Například fialová barva, tedy `RGB(255,0,255)` se uloží do proměnné `ColorA` pomocí

```
ColorA = RGB(255,0,255)
```

1.1.2 Deklarace proměnných

V RS nehrají deklarace proměnných nijak významnou roli - program deklaruje proměnné automaticky a druh proměnné určí podle toho, co je do proměnné právě přiřazováno. Deklarace proměnné slouží jen k tomu, aby překladač ověřil, jestli nebyla nějaká proměnná omylem deklarována (a použita) pro více různých účelů. Deklarace proměnné A se pak napíše jako `Dim A`.

1.2 Příkazy

1.2.1 Seznam příkazů

Seznam všech použitých příkazů RS se nachází na konci práce. Kompletní přehled příkazů Rhinoscriptu včetně nápovědy viz [1].

1.2.2 Volání příkazů

Volat příkazy Rhina je v RS možné dvojím způsobem. Základní a uživatelsky příznivý způsob je vybrat jednu z předdefinovaných funkcí. Například úsečka z bodu $\mathbf{A} = (0, 0, 0)$ do bodu $\mathbf{B} = (0, 0, 1)$, jejíž identifikátor se uloží do proměnné `linAB`, se vytvoří pomocí následujícího kódu.

```
1  Dim arrA, arrB, linAB
2  arrA  = Array(0,0,0)
3  arrB  = Array(0,0,1)
4  linAB = Rhino.AddLine(arrA, arrB)
```

Kód 1.1: Tvorba úsečky pomocí předdefinované funkce

Byl tedy zavolán příkaz Rhina, který vytvoří úsečku mezi body danými souřadnicemi, které odpovídají hodnotám uloženým v polích `arrA` a `arrB`. Je důležité upozornit, že zdaleka ne všechny příkazy, které jsou k dispozici v Rhinu, jsou takto připraveny k pohodlnému užití. Pokud je třeba takové příkazy využít, musí být buď nahrazeny matematicky (např. analytický výpočet bodů tečny a použití substruktury `AddLine` místo příkazu tečna ke křivce) nebo je lze z Rhina zavolat vstoupením na příkazovou řádku Rhina. To umožňuje příkaz `Rhino.Command`, jehož argumentem je textový řetězec obsahující název příkazu Rhina a poté jeho argumenty oddělené mezerami. Tvorba úsečky jako v kódu 1.1 tedy vypadá následovně:

```
1    Dim arrA, arrB, arrC
2    Rhino.Command "_Line 0,0,0 0,0,1"
3    linAB = Rhino.FirstObject
```

Kód 1.2: Tvorba úsečky pomocí vstupu na příkazovou řádku Rhina 1

Tento způsob je sice universální (lze použít jakýkoli příkaz Rhina), ale v mnohém těžkopádný. Za prvé je potřeba dodatečně uložit identifikátor úsečky do proměnné `linAB` pomocí příkazu `FirstObject`, za druhé ve chvíli, kdy jsou (jako u kódu 1.1) souřadnice bodů uloženy v samostatných proměnných, stává se tento způsob velmi nepřehledným:

```
1    Dim arrA, arrB, linAB
2    arrA = Array(0,0,0)
3    arrB = Array(0,0,1)
4    Rhino.Command "_Line " & arrA(0) & "," & arrA(1) & "," &
       arrA(2) & " " & arrB(0) & "," & arrB(1) & "," & arrB
      (2)
5    linAB = Rhino.FirstObject
```

Kód 1.3: Tvorba úsečky pomocí vstupu na příkazovou řádku Rhina 2

Pokud je tedy zapotřebí opakovaně volat nějaký příkaz, který nemá předdefinovanou funkci, je nejlépe vytvořit si funkci vlastní.

1.2.3 Vytváření a volání funkcí

Funkce je série příkazů, které RS provede v určeném pořadí po sobě, použije k tomu zadané argumenty a posléze může (a nemusí) vrátit určitou hodnotu. To je užitečné, pokud je činnost v programu opakována a to zejména pokud je složitá (např. kód 1.3).

Následuje příklad tvorby a zavolání funkce `MyLine`, která je identická s funkcí `Rhino.AddLine`.

```
1    Function MyLine (arrPoint1, arrPoint2)
2    Rhino.Command "_Line " & arrPoint1(0) & "," & arrPoint1
      (1) & "," & arrPoint1(2) & " " & arrPoint2(0) & "," &
      arrPoint2(1) & "," & arrPoint2(2)
3    MyLine = Rhino.FirstObject
4    End Function
5
6    Dim arrA, arrB, linAB
```

```
7    arrA  = Array(0,0,0)
8    arrB  = Array(0,0,1)
9    linAB = MyLine (arrA, arrB)
```

Kód 1.4: Tvorba a volání funkce pro tvorbu úsečky

Velmi podobně, jako funkce, fungují subrutiny, jen neumí vracet hodnotu. To je možné obejít tím, že proměnná, do níž má být uložena hodnota, kterou funkce vrací, je jedením z argumentů funkce. Stejného efektu jako v kódu 1.4 lze dosáhnout následujícím způsobem.

```
1    Sub MyLine (arrPoint1, arrPoint2, strName)
2    Rhino.Command "_Line " & arrPoint1(0) & "," & arrPoint1
      (1) & "," & arrPoint1(2) & " " & arrPoint2(0) & "," &
      arrPoint2(1) & "," & arrPoint2(2)
3    strName = FirstObject
4    End Sub
5
6    Dim arrA, arrB, linAB
7    arrA = Array(0,0,0)
8    arrB = Array(0,0,1)
9    MyLine arrA, arrB, linAB
```

Kód 1.5: Tvorba a volání subrutiny pro tvorbu úsečky

1.2.4 Cykly

V této práci byly využity jen „for“ cykly, tedy cykly, které se nasazují, je-li třeba nějakou činnost (např. funkci `Deed`) opakovat n krát.

```
1    For i = 1 To n
2      Deed
3    Next
```

Kód 1.6: For cyklus v RhinoScriptu

1.2.5 Podmínky

Pokud se některá část programu (např. opět funkce `Deed`) má spustit pouze za splnění určité podmínky (např. $n > 0$) lze toho dosáhnout následujícím kódem.

```
1    If n>0 Then
2      Deed
3    End If
```

Kód 1.7: Podmínka If v RhinoScriptu

2. Rotor

Následuje text popisující tvorbu rotoru pomocí RS.

2.1 Lopatkový kanál

Nyní bude popsána úvodní část skriptu a dále část skriptu tvořící ohraničení lopatkového kanálu.

Zadávání parametrů

Na úvod skriptu jsou zavedeny hodnoty proměnných, které může uživatel měnit.

```
R_hR0 = 19
```

Kód 2.1: Zadávání parametrů modelu

V souladu s tabulkou 1.1 je hodnota parametru h_{R0} předcházena R, protože se jedná o reálné číslo. Obdobně jsou v souladu s tabulkou 1.1 a se seznamem parametrů na konci práce zadány ostatní parametry. Na tomto místě skriptu je také možno přepisovat hodnoty jednotlivých parametrů.

Tabulka 2.1: Pojmenování geometrie ve skriptu

geom. útvar	příklad	
	název	název v kódu
úsečka A_RB_R	`linAR_BR`	
přímka p daná směrovým vektorem	`V_p`	
oblouk o_S	`arcS`	
křivka k_{R1}	`crvR1`	
plocha	π_{R1}	`srfR1`
	π_{B1}	`srfB1`

Tvorba bodů a křivek

V další části skriptu jsou popsány jednotlivé body a vytvořeny úsečky popisující ohraničení kanálu. Body jsou popsány tříprvkovými poli reálných čísel a úsečky svými identifikátory. Následuje ukázka kódu popisující dva body a úsečku mezi nimi.

```
1  Dim A_AR, A_BR
2  Dim linAR_BR
3  A_AR = Array (R_hR0+R_hR2, R_dR1/2, 0)
4  A_BR = Array (R_hR0+R_hR2-R_hR3, R_dR1/2, 0)
5  linAR_BR = Rhino.AddLine (A_AR, A_BR)
```

Kód 2.2: Tvorba bodů a úsečky

Proměnná, do níž se ukládá identifikátor úsečky A_RB_R, byla v programu pojmenována pro přehlednost `linAR_BR`, podobně i ostatní úsečky. Tabulka 2.1 shrnuje tyto změny názvů pro jednotlivé typy geometrických útvarů. Křivky jsou následně

vytvořeny pomocí následujícího příkazu `AddCurve`, který přidá Beziérovu křivku dle zadaného stupně a řídicích bodů (v případě následujícího kódu křivky druhého a třetího stupně).

```
Dim CrvR1, CrvRA
Dim A_CrvR1_Points, A_CrvRA_Points
A_CrvR1_Points = Array (A_BR, A_DR0, A_ER)
A_CrvR2_Points = Array (A_FR, A_CR0, A_LR0, A_GR)
CrvR1 = Rhino.AddCurve (A_CrvR1_Points, 2)
CrvR2 = Rhino.AddCurve (A_CrvRA_Points, 3)
```

Kód 2.3: Tvorba křivek

Způsobem ukázaným v kódech 2.2 a 2.3 byly vytvořeny všechny křivky a úsečky z obrázku 2.6 (viz práce) v souladu s tabulkami 2.1 a 2.2 (viz práce).

Tvorba ploch

Potřebné rotační plochy byl vytvořeny ve dvou krocích - nejprve spojením příslušných křivek do jedné příkazem `JoinCurves` a posléze použitím příkazu `AddRevSrf`. Tento příkaz přidá rotační plochu na základě zadané tvořící křivky, osy rotace a počátečního a koncového úhlu.

```
Dim A_Axis, A_MR, A_NR, joiTR_R1, joiUR_FR, joiCUT,
 srfR1, srfR2, srfRCut
joiTR_R1 = Rhino.JoinCurves(Array(linTR_AR, linAR_BR,
 CrvR1),    0) (0)
joiUR_FR = Rhino.JoinCurves(Array(linHR_UR, linHR_GR,
 CrvRA), 0) (0)
A_MR    = Array(1, 0, 0)
A_NR    = Array(0, 0, 0)
A_Axis  = Array(A_MR, A_NR)
srfR1   = Rhino.AddRevSrf(joiTR_R1, A_Axis, -360/2/I_nR,
  360/2/I_nR)
srfR2   = Rhino.AddRevSrf(joiUR_FR, A_Axis, -360/2/I_nR,
  360/2/I_nR)
srfRCut = Rhino.AddRevSrf(joiCUT,   A_Axis, -90, 90)
```

Kód 2.4: Tvorba ploch rotorového kanálu

2.2 Plocha rotorové lopatky

Následuje popis skriptu tvořící rotorové lopatky.

Body, úsečky a křivky

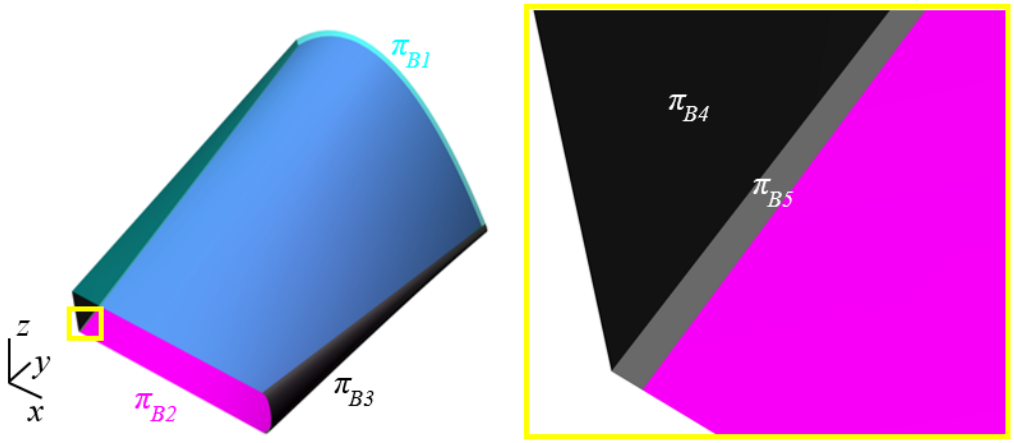
Po vytvoření lopatkového kanálu následuje ve skriptu zadávání bodů použitých pro tvorbu lopatky samotné, tedy bodů z tabulky 2.6 (viz práce). Výpočet souřadnic bodů a jejich ukládání do proměnných bylo provedeno stejně, jako v případě kódu 2.2. Úsečky a křivky dle tabulky 2.7 (viz práce) se na základě bodů z tabulky 2.6 (viz práce) vytvoří obdobně, jako tomu bylo u křivek ohraničení rotorového kanálu (kódy 2.2 a 2.3).

Plochy

Jak bylo uvedeno, plochy rotorových lopatek jsou přímkové přechodové plochy. Takovou plochu je možno přidat příkazem `AddEdgeSrf`, jehož argument je pole hraničních křivek (pro dvě křivky vytvoří přímkovou přechodovou plochu, pro 3 a 4 křivky Coonsovu Bilineární plochu [2]). Tyto plochy byly následně spojeny pomocí příkazu `JoinSurfaces` (argumenty tohoto příkazu jsou pole identifikátorů ploch, které mají být spojeny, a hodnota True/False pro vymazání/nevymazání původních ploch). Plochy z tabulky 2.8 (viz práce) byly vytvořeny následujícím způsobem. Hotové plochy jsou na obrázku 2.1.

```
Dim srfB1, srfB2, srfB3, srfB4, srfB5, A_Blade, srfBlade
srfB1 = Rhino.AddEdgeSrf (Array(crvB1, linOR_PR))
srfB2 = Rhino.AddEdgeSrf (Array(crvB2, linSR_RR))
srfB3 = Rhino.AddEdgeSrf (Array(crvB3, crvB4))
srfB4 = Rhino.AddEdgeSrf (Array(linRR_End20,
 linVR2_End11))
srfB5 = Rhino.AddEdgeSrf (Array(linBLD2, linOR_VR2))
A_Blade = Array (srfB1, srfB3, srfB2, srfB4, srfB5)
srfB  = Rhino.JoinSurfaces (A_Blade, False)
```

Kód 2.5: Tvorba ploch rotorových lopatek



Obrázek 2.1: Plochy rotorové lopatky

2.3 Dokončení rotoru

Popsané plochy ohraničující plochu lopatkového kanálu a rotorové lopatky bylo potřeba oříznout (anglicky „trim“), okopírovat kruhovým polem a dále dodělat střed rotoru.

2.3.1 Střed rotoru

V původním modelu je hřídel rotoru zakončen komplikovanou tvarovou plochou (obrázek 2.2). Tato plocha nemá velký význam na proudění v rotoru a proto nemá smysl tuto plochu přesným způsobem aproximovat. Na místo této plochy byla v novém modelu umístěna polovina kulové plochy. Nejprve byla vytvořena

kulová plocha (příkaz `AddSphere`, jehož argumenty jsou středový bod a poloměr), poté řezná rovina (ta byla vytvořena příkazem `AddCutPlane`). Nakonec proběhlo rozpůlení koule pomocí řezné roviny a

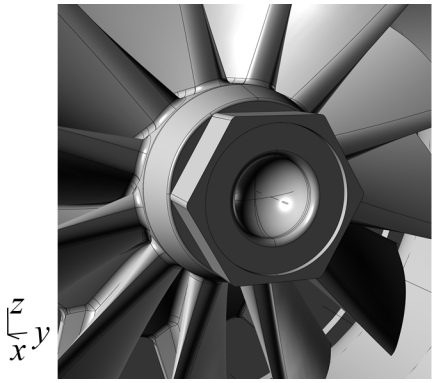
```
Dim srfSphere, srfCutPlane, A_HalfSpheres

srfSphere = Rhino.AddSphere ( Array(A_UR(0),0,0), R_dR0/2)

srfCutPlane = Rhino.AddCutPlane (Array(srfSphere), Array(
    A_UR(0),2*A_UR(1),0), Array(A_UR(0),-2*A_UR(1),0) )
A_HalfSpheres = Rhino.SplitBrep (srfSphere, srfCutPlane,
    True) 'delete input

Rhino.DeleteObject srfCutPlane
Rhino.DeleteObject A_HalfSpheres (1)
```

Kód 2.6: Tvorba kulové plochy na konci rotorového hřídele

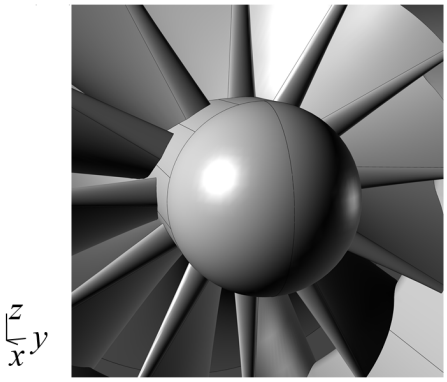
Nový střed rotoru je na obrázku 2.3.

2.3.2 Oříznutí ploch

Výpočet oříznutí ploch je úloha přesahující svým rozsahem a náročností obsah této práce i náplň bakalářského studia. Proto bude tato část výpočtu provedena pouze v Rhinu.



Obrázek 2.2: Střed rotoru původního modelu

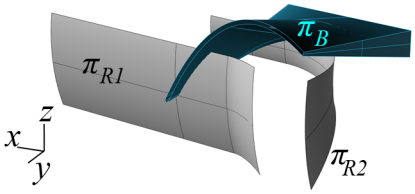


Obrázek 2.3: Střed rotoru nového modelu

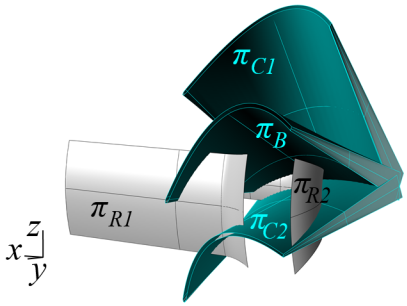
Tvorba pomocných ploch

Na obrázku 2.4 jsou plochy, které byly dosud vytvořeny, tedy plochy ohraničující lopatkový kanál (π_{R1} a π_{R2}) a plocha rotorové lopatky (π_B). Je vidět, že tyto plochy se navzájem neořežou dostatečně. Proto byly k ořezání použity pomocné plochy π_{C1} a π_{C2} (obrázek 2.5) a $\pi_R Cut$ je zobrazena na obrázku 2.7).

Pomocné plochy π_{C1} a π_{C2} byly vytvořeny pomocí kódu 2.7. Použity byly příkazy `XformRotation`, který vytvoří transformační matici pro rotaci. Parametry tohoto příkazu jsou úhel otočení, osa otáčení a střed otáčení. Pro rotaci požadovanou v kódu 2.7 (tedy otočení kolem osy x o úhel $\phi = \frac{360}{n_R}$) by matice vypadala



Obrázek 2.4: Rotorové plochy k ořezání



Obrázek 2.5: Pomocné plochy π_{C1} a π_{C2}

následujícím způsobem:

$$\mathbf{T}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \text{ [3].} \tag{2.1}$$

Samotné otočení bylo provedeno příkazem `TransformObject`, který transformuje objekt na základě transformační matice.

```
Dim A_Transformation1, A_Transformation2, R_Angle,
    A_Center, A_Cut0, A_Cut1, A_Cut2
Dim srfCut1, srfCut2
Dim V_Axis

R_Angle = 360/I_nR
A_Center = Array (0,0,0)
V_Axis = Array (1,0,0)

A_Transformation1 = Rhino.XformRotation(R_Angle, V_Axis,
    A_Center)
srfC1 = Rhino.TransformObject (srfB, A_Transformation1,
    True)
A_Transformation2 = Rhino.XformRotation(-R_Angle, V_Axis,
    A_Center)
srfC2 = Rhino.TransformObject (srfB, A_Transformation2,
    True)
```

Kód 2.7: Tvorba pomocných ploch ploch pro oříznutí lopatkového kanálu

Oříznutí lopatkového kanálu

Plochy π_{R1} a π_{R2} byly oříznuty pomocí ploch π_B, π_{C1} a π_{C2}. Byl k tomu použit příkaz `SplitBrep`, který rozdělí plochu pomocí střihacího objektu a vrátí pole identifikátorů nových ploch, které tímto dělením vzniknou. V kódu byly pak příkazem `DeleteObject` vymazány ty plochy, které nebyly dále potřeba. Výsledné ořezané plochy jsou poté vždy uloženy do nové proměnné.

Nejprve byly plochy ořezány plochou π_B. Kód je pro ořezání obou ploch obdobný, proto je uvedena jen část kódu zajišťující ořezání plochy π_{R1}.

```
Dim A_SplittedR1_0

```

```
A_SplittedR1_0 = Rhino.SplitBrep (srfR1, srfC0, True)
Rhino.DeleteObject A_SplittedR1_0 (0)
srfR1 = A_SplittedR1_0(1)
```

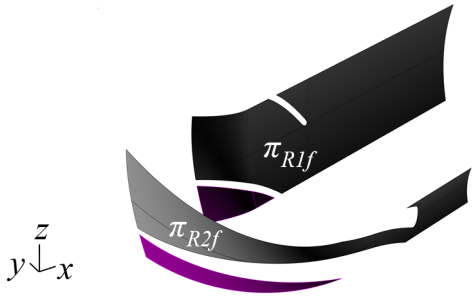
Kód 2.8: Ořezávání plochy π_{R1} plochou π_B

Pro ořezání pomocnými plochami π_{C1} a π_{C2} bylo zapotřebí složitějšího algoritmu. Nejprve bylo zjištěno, zda pro danou geometrii existuje průsečík ploch π_{R1} a π_{C1}, tedy jestli příkaz `SplitBrep` vrátil prázdné pole (`Null`). Pokud ne, byla oříznutá plocha uložena do původní proměnné a vymazána přebytečná plocha. V případě vzniku tří ploch by příkaz `SplitBrep` vrátil pole tří prvků. To bylo zkontrolováno pomocí funkce `GetArrayDim`, která vrátí počet prvků daného pole. Pokud tedy vznikla třetí plocha, byla uložena do nové proměnné. Opět je uveden jen kód pro oříznutí plochy π_{R1}.

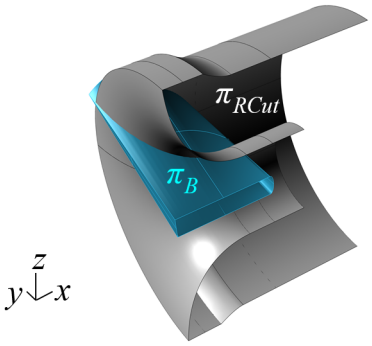
```
Dim A_SplittedR1_1, A_SplittedR1_2
Dim srfRotorEnd1, srfRotorEnd2, A_BladeEnd1, A_BladeEnd2

A_SplittedR1_1 = Rhino.SplitBrep (srfR1, srfC1, True)
If Not IsNull (A_SplittedR1_1) Then
  Rhino.DeleteObject A_SplittedR1_1(0)
  srfR1 = A_SplittedR1_1(1)
  If GetArrayDim(A_SplittedR1_1) > 2 Then
    srfRotorEnd1 = A_SplittedR1_1 (2)
  End If
End If

A_SplittedR1_2 = Rhino.SplitBrep (srfR1, srfC2, True)
If Not IsNull (A_SplittedR1_2) Then
  srfRotorEnd = A_SplittedR1_2(1)
  Rhino.DeleteObject A_SplittedR1_2 (0)
  If GetArrayDim(A_SplittedR1_2) > 2 Then
    srfR1 = A_SplittedR1_2 (2)
  End If
End If
```

Kód 2.9: Ořezávání plochy π_{R1} plochou π_{C1}

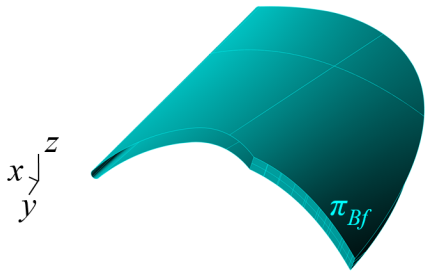
Oříznuté plochy π_{R1f} a π_{R2f} po ořezání jsou na obrázku 2.6.



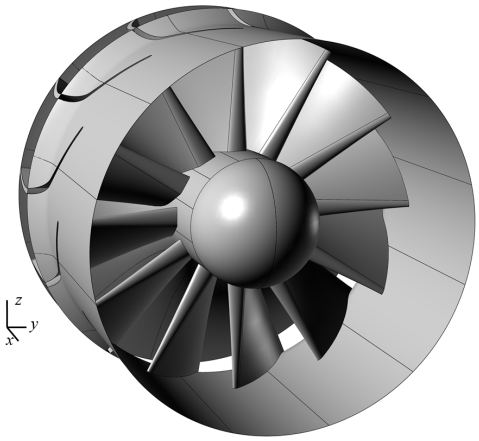
Obrázek 2.6: Oříznuté plochy π_{R1f} a π_{R2f}



Obrázek 2.7: Pomocná plocha π_{RCut}



Obrázek 2.8:



Obrázek 2.9: Náhradní model rotoru

Oříznutí lopatky

Plocha π_B byla ořezána plochou π_{RCut} (plochy jsou na obrázku 2.7). Kód, je v principu stejný s kódy 2.8 a 2.9, jen je pro odstraňování přebytečných ploch použit „for“ cyklus.

```
Dim A_SplittedBlade

A_SplittedBlade = Rhino.SplitBrep (srfB, srfRCut, 1)

For i = 1 To GetArrayDim (A_SplittedBlade) - 1
    Rhino.DeleteObject A_SplittedBlade (i)
Next

srfB = A_SplittedBlade (0)
```

Kód 2.10: Ořezávání plochy π_B plochou π_{RCut}

Tuto oříznutou lopatku bylo ještě třeba z jedné strany uzavřít. Proto byly nalezeny křivky odpovídající hranám této plochy (příkaz `DuplicateEdgeCurves`), vhodné z těchto křivek byly spojeny a mezi nimi byla vytvořena přímková přechodová plocha. Tato plocha byla posléze sjednocena s plochou π_B a nepoužité hraniční křivky byly smazány. Hotová ořezaná lopatka π_{Bf} je na obrázku 2.8.

```
Dim crvEdge1, crvEdge2, srfB8, A_DuplicatedEdges, A_Edge1

A_DuplicatedEdges = Rhino.DuplicateEdgeCurves (srfB)
Rhino.DeleteObject A_DuplicatedEdges (0)
Rhino.DeleteObject A_DuplicatedEdges (1)
Rhino.DeleteObject A_DuplicatedEdges (3)
Rhino.DeleteObject A_DuplicatedEdges (4)
Rhino.DeleteObject A_DuplicatedEdges (5)
Rhino.DeleteObject A_DuplicatedEdges (6)
Rhino.DeleteObject A_DuplicatedEdges (8)
Rhino.DeleteObject A_DuplicatedEdges (10)

A_Edge1 = Array (A_DuplicatedEdges (2), A_DuplicatedEdges
   (9) )
crvEdge1 = Rhino.JoinCurves (A_Edge1, 1) (0)
```

```
15 crvEdge2 = A_DuplicatedEdges (7)
16
17 srfB8 = Rhino.AddEdgeSrf (Array (crvEdge1, crvEdge2) )
18 srfB  = Rhino.JoinSurfaces (Array (srfB, srfB8), 1)
```

Kód 2.11: Náběžná hrana rotoru

2.3.3 Kruhové pole rotoru

Na závěr byly oříznuté plochy lopatky a lopatkového kanálu n_R-krát zkopírovány kruhovým polem. Protože RS nemá předdefinovanou subrutinu pro kruhové pole, bylo nejprve třeba ji vytvořit pomocí příkazů `XformRotation` a `TransformObject`, podmínky a cyklu (tyto příkazy již byly použity v kódu 2.7 na str. 12).

```
1  Sub MyArrayPolarX (arrObjects, intCount, arrCenter) 'does
       a polar array around the x axis
2    Dim realAngle, arrAxis, arrXform, i
3
4    If IsNull(arrObjects) Then Exit Sub
5    If IsNull(intCount) Then Exit Sub
6    If IsNull(arrCenter) Then Exit Sub
7
8    realAngle = 360.0 / intCount
9
10   For i = 1 To intCount - 1
11     arrAxis = Array(1,0,0) ' world x-axis
12     arrXform = Rhino.XformRotation(realAngle * i, arrAxis,
       arrCenter)
13       Rhino.TransformObjects arrObjects, arrXform, True
14     Next
15 End Sub
```

Kód 2.12: Subrutina pro kruhové pole

Samotná aplikace kruhového pole pak proběhla pro plochy π_{R1f}, π_{R2f}, π_{Bf} dle kódu 2.13. Kruhové pole bylo aplikováno i na oddělené části ploch π_{R1f} a π_{R2f} (na obrázku 2.6 označeny fialově). Existence těchto oddělených ploch byla ověřena stejnými podmínkami, jaké byly ukázány v kódu 2.9 (v následujícím kódu ukázáno pro jednu část plochy π_R).

```
1 If Not IsNull (A_SplittedR1_1) Then
2   If GetArrayDim(A_SplittedR1_1) > 2 Then
3     MyArrayPolarX Array(srfRotorEnd1), I_nR, Array (1,0,0)
4   End If
5 End If
6
7 MyArrayPolarX Array(srfR1), I_nR, Array (1,0,0)
8 MyArrayPolarX Array(srfR2), I_nR, Array (1,0,0)
```

Kód 2.13: Kruhové pole rotoru

Celý rotor po použití kruhového pole je na obrázku 2.9.

3. Stator

3.1 Statorové lopatky

Tato část skriptu je v mnohém podobná, jako část skriptu popisující rotor, proto je zde často odkazováno do kapitoly 2 a detailně je popsáno jen použití příkazů, které se dosud neobjevily.

Tvořicí křivky

Část skriptu popisující tvorbu statoru začíná výpočtem souřadnic bodů dle tabulky uvedené v halvním souboru a jejich ukládáním do proměnných. Kód je v této části obdobou kódu 2.2 [1]. Křivky k_{S1} a k_{S2} byly modelovány pomocí příkazu `AddCurve` [2].

Oblouk o_R byl modelován příkazem `AddArc3Pt`. Argumentem tohoto příkazu je pole tří bodů a tento příkaz přidá oblouk procházející třemi bodu v takovém pořadí, v jakém jsou body v poli umístěny.

```
1    Dim ArcS
2    ArcS = Rhino.AddArc3Pt (A_AS, A_BS, A_ES1)
```

Kód 3.1: Tvorba oblouku pomocí 3 bodů

Plocha

Poté, co jsou tvořicí křivky vytvořeny, jsou spojeny v jednu (s názvem `joiCrvS`) příkazem `JoinCurves`[3]. Ze spojené křivky je poté vytvořena plocha lopatky příkazem `ExtrudeCurveStraight`. Argumenty tohoto příkazu jsou identifikátor křivky, jejímž vysunutím má plocha vzniknout, a dva body určující počátek a konec dráhy posuvného pohybu. Tento posuvný pohyb se koná kolmo na rovinu, v níž vytahovaná křivka leží.

```
1    Dim A_ExtStart, A_ExtEnd
2    A_ExtStart = Array(-2 + A_SS1(0)        , A_SS1(1), A_SS1
      (2) )
3    A_ExtEnd   = Array( 2 + A_SS1(0) + R_hS, A_SS1(1), A_SS1
      (2) )
4
5    Dim srfBldS
6    srfBldS = Rhino.ExtrudeCurveStraight (joiCrvS,
      A_ExtStart, A_ExtEnd )
```

Kód 3.2: Tvorba plochy vytažením

Souřadnice x bodů `A_ExtStart` a `A_ExtEnd` jsou posunuty, aby plocha π_{BS} přesahovala na obě strany hraničních ploch statoru π_{S4} a π_{S5} a bylo tak zajištěno, že Rhino dokáže najít průsečíky (viz obrázek 3.1).

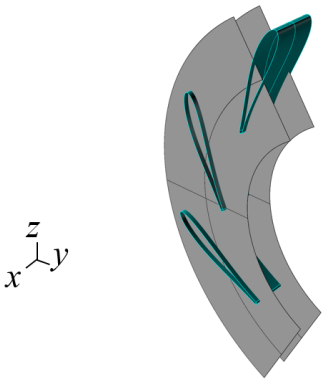
[1] str. 8
[2] viz kód 2.3 (str. 9)
[3] viz kód 2.4 (str. 9)

3.2 Ohraničení statoru

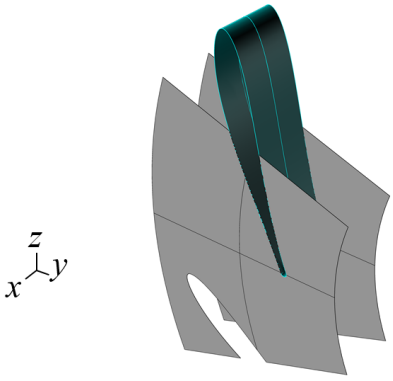
Tvorba úseček příkazem `AddLine` a rotačních ploch příkaze `AddRevSrf` byla již popsána v předchozích částech práce[4].

3.3 Dokončení statoru

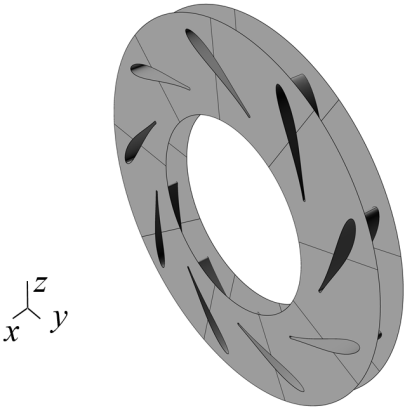
Nakonec byly vytvořené prvky statoru ořezány a okopírovány kruhovým polem. Princip tohoto algoritmu je shodný s postupem použitým pro rotor (viz kapitola 2.3), kde byl tento algoritmus podrobně popsán. Proto je zde uvedeno stručné shrnutí pro stator. Na obrázku 3.1 jsou vidět plochy lopatky i ohraničení připravené k ořezání a na obrázku 3.2 jsou již ořezané plochy připravené ke kopírování kruhovým polem. Hotový stator po aplikování kruhového pole je vidět na obrázku 3.3 a na obrázku 3.4 je kompletní náhradní model včetně rotoru.



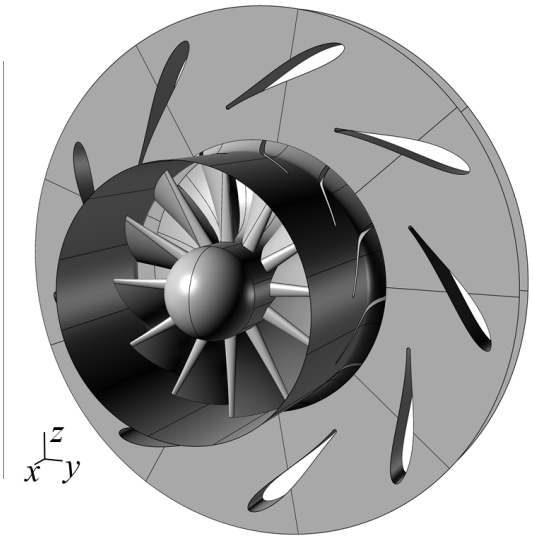
Obrázek 3.1: Plochy statoru připravené na ořezání



Obrázek 3.2: Ořezané plochy statoru



Obrázek 3.3: Náhradní model statoru



Obrázek 3.4: Náhradní model radiální turbíny

[4] viz kódy 2.2 (str. 8) a 2.4 (str. 9)

Seznam použité literatury

[1] RhinoScript. *Rhinoceros* [online]. Roger de Flor, 32-34,Barcelona, 08018 Spain: Robert McNeel & Associates, 2017 [cit. 2017-06-02]. Dostupné z: http://developer.rhino3d.com/api/rhinoscript/index.html.

[2] LINKEOVÁ, Ivana. *Základy počítačového modelování křivek a ploch.* Praha, Nakladatelství ČVUT, 2008. ISBN 978-80-01-04011-9.

[3] VALÁŠEK, Michael, Zbyněk ŠIKA a Václav BAUMA. *Mechanika B.* V Praze: Vydavatelství ČVUT, 2004. ISBN 80-010-2919-0.

[4] Including Scripts. *Rhino Developer Docs* [online]. Roger de Flor, 32-34, bajos Barcelona, 08018 Spain: Robert McNeel & Associates, 2017 [cit. 2017-06-02]. Dostupné z: http://developer.rhino3d.com/guides/rhinoscript/including-scripts/.

Přílohy

Seznam použitých příkazů z programu Rhinoceros

_Angle Vyhodnotí úhel sevřený dvěma orientovanými úsečkami.

_CrvDeviation Vyhodnotí maximální odchylku dvou křivek.

_List Vypíše základní vlastnosti objektu (bodu, křivky, plochy) - mj. stupeň, souřadnice řídicích bodů a jejich počet a počet uzlů řídicí sítě.

_MeanCurve Vytvoří střední křivku mezi dvěma otevřenými nebo uzavřenými křivkami.

_Offset Vytvoří ekvidistantní křivku k zadané rovinné křivce.

_Rotate Otočí dané objekty okolo zadaného středu o zadaný úhel nebo o úhel odpovídající úhlové rozteči dvou referenčních bodů.

_Section Vytvoří křivku zadanou jako průnik plochy a roviny kolmé na jednu z rovin xy, yz, xz.

_Untrim Zruší střih vybrané stříhané plochy.

_Zebra Vykreslí na vybraných plochách zebří pruhy pro analýzu spojitosti těchto ploch.

Seznam použitých příkazů z jazyka RhinoScript

AddArc3Pt přidá oblouk procházející 3 body v takovém pořadí, v jakém jsou body zadány, viz kód

AddCurve přidá Beziérovu křivku podle argumentů (pole řídicích bodů, stupeň křivky), příklad viz kód 2.3, s. 9

AddCutPlane přidá řeznou rovinu, příklad viz kód 2.6, s. 11

AddLine přidá úsečku podle argumentů (dva body), příklad viz kód 2.2, s. 8

AddEdgeSrf přidá plochu podle argumentů (přímkovou přechodovou plochu, pokud jsou dány 2 křivky, Coonsovu bilineární plochu, pokud jsou zadány 3 nebo 4 křivky), příklad viz kód

AddSphere přidá kulovou plochu, argumenty jsou středový bod a poloměr, příklad viz kód 2.6, s. 11

AddRevSrf přidá rotační plochu podle argumentů (identifikátor křivky, osa rotace, počáteční úhel, konečný úhel), příklad viz kód 2.2, s. 8

DeleteObject odstraní objekt, argumentem je pole identifikátorů objektů, které mají být smazány, viz kód 2.2, s. 8

DuplicateEdgeCurves duplikuje hraniční křivky dané plochy, viz kód 2.11, s. 14

ExtrudeCurveStraight přidá plochu posunutím výtvarné křivky ve směru kolmém na rovinu této křiky

FirstObject vrátí identifikátor naposledy vytvořeného objektu, příklad viz kód 1.2, s. 6

JoinCurves spojí zadané křivky do (pokud možno) jedné, argumentem je pole identifikátorů křivek a hodnota True/False pro smazání/nesmazání původních křivek, příklad viz kód 2.4, s. 9

JoinSurfaces spojí zadané plochy do jedné (pokud možno), argumenty jsou pole identifikátorů ploch a hodnota True/False pro smazání/nesmazání původních ploch, příklad viz kód 2.5, s. 10

SplitBrep rozdělí plochu na více ploch pomocí stříhacího objektu, argumenty jsou identifikátor stříhané plochy, identifikátor stříhacího objektu a True/False pro zachování/smazání původního objektu

TransformObject transformuje objekt pomocí transformační matice, argumenty jsou identifikátor zadaného objektu, transformační matice a True/False pro zachování/vymazání původního objektu, příklad viz kód 2.7, s. 12

XformRotation vytvoří transformační matici pro rotaci, argumenty jsou úhel, směrový vektor osy a střed otáčení, příklad viz kód 2.7, s. 12