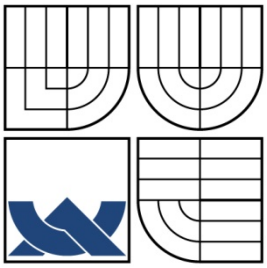
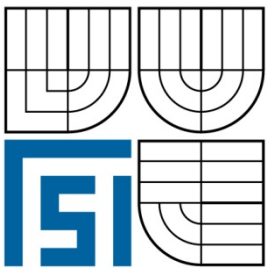


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Brno University of Technology



FAKULTA STROJNÍHO INŽENÝRSTVÍ
Faculty of Mechanical Engineering

ÚSTAV DOPRAVNÍ TECHNIKY
Institute of Automotive Engineering

MATEMATICKÉ MODELOVÁNÍ MECHANICKÝCH SYSTÉMŮ VOZIDEL

DIPLOMOVÁ PRÁCE
Diploma Thesis

AUTOR PRÁCE — Lubor Zháňal
Author

VEDOUCÍ PRÁCE — Ing. Petr Porteš Dr.
Supervisor

BRNO 2008

ANOTACE

Diplomová práce se zabývá návrhem a realizací výpočetního systému, který by umožňoval interaktivní simulace kinematiky a dynamiky mechanických systémů vozidel či obecných mechanismů za pomoci moderní výpočetní techniky. Rozebrán je základní princip činnosti, optimalizace numerických parametrů simulace, stejně jako samotná implementace do podoby funkční aplikace a její stěžejní části. Popsány jsou také možnosti využití prostředků virtuální reality ve spojitosti s vizualizací simulačního procesu. Závěrečné srovnávací testy potvrzují funkčnost a vysokou výkonnost vytvořeného systému.

KLÍČOVÁ SLOVA

Mechanismy, simulace, multi-body systémy, interaktivita, virtuální realita.

ANNOTATION

This thesis deals with design and implementation of the computing system, which would allow interactive simulation of kinematics and dynamics of vehicle mechanical systems or general mechanisms with the help of modern computer technology. In the thesis is described a basic working principle, optimization of numerical simulation parameters, as well as the implementation itself in the form of functional application and its key parts. Described are also possibilities for the use of virtual reality devices in conjunction with visualization of simulation process. Final comparative tests have confirmed the functionality and high performance of the created system.

KEY WORDS

Mechanisms, simulation, multi-body systems, interactivity, virtual reality.

BIBLIOGRAFICKÁ CITACE

ZHÁŇAL, L. Matematické modelování mechanických systémů vozidel. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2008. 78 s. Vedoucí diplomové práce Ing. Petr Porteš, Dr.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s využitím uvedených zdrojů.

PODĚKOVÁNÍ

Poděkování za podporu při psaní této diplomové práce náleží mé rodině, jež mi byla nenahraditelným zázemím, mému vedoucímu Ing. Petru Portešovi Dr., jenž mi byl rádcem a in memoriam Zuzaně Navarové d.T., jejíž dílo doprovází mé myšlenky.

„...jen vůně kardamomu
a ráno domů, domů...“

OBSAH

1. Úvod 10
1.1. Multi-body systémy 11
1.1.1. Rozdělení multi-body systémů 11
1.1.2. Moderní trendy 11
1.2. Multi-body systém Mechanics 12
2. Metoda korekčních sil a její využití 13
2.1. Základní charakteristika metody 13
2.2. Princip MKS 14
3. Elementární prvky MKS 16
3.1. Typy mechanismů 16
3.2. Jednotky 16
3.3. Základní elementy 16
3.3.1. Tělesa (Bodies) 17
3.3.2. Vazby (Binds) 18
3.3.3 Linky (Links) 18
3.3.4. Síly (Forces) 19
4. Postup řešení 20
4.1. Výpočetní schéma 20
4.2. Výpočet pohybu těles 21
4.2.1. Určení silových a momentových výslednic k těžišti tělesa 21
4.2.2. Aktualizace kinematického stavu těžiště tělesa 22
4.2.3. Aktualizace kinematického stavu dalších bodů 22
4.3. Výpočet korekčních sil vazeb 23
4.3.1. Určení diferenčních vektorů kinematických veličin vázaných bodů 23
4.3.2. Výpočet korekčních sil 24
4.3.3. Aplikace výsledných sil do vázaných bodů 24
4.4. Výpočet reakčních sil linků 25
4.5. Výpočet vnějšího silového působení 26
5. Optimalizace vazebných parametrů 27
5.1. Stabilita simulace 27

5.2. Popis parametrů 27
5.3. Vliv přenosu na typ vazby 28
5.4. Vliv parametrů tvrdé vazby 29
5.5. Vliv parametrů pružné vazby 30
5.6. Vliv časového kroku na vazby 30
5.7. Zhodnocení tvrdé a pružné vazby 31
5.8. Předdefinované vazby 32
5.8.1. Typy předdefinovaných vazeb 32
5.8.2. Automatický odhad parametrů 33
6. Programová implementace MKS 35
6.1. Vývojové prostředí 35
6.2. Struktura aplikace 36
6.3. Parametrický strom 37
6.3.1. Datové typy 37
6.3.2. Struktura stromu 38
6.3.3. Zásobník stavů 39
6.4. Optimalizace jádra 39
6.4.1. SIMD 39
6.4.2. Paralelizace výpočtů 40
6.4.3. GP-GPU 41
6.5. Grafický subsystém 42
6.5.1. Import geometrie z CAD systémů 43
6.6. Konzole a skriptování 44
6.6.1. Konzole 44
6.6.2. Skripty 45
6.7. Rozšiřující moduly 46
6.7.1. Integrovaný průvodce programováním modulů 47
6.8. Možnosti běhu pod Linuxem 47
7. Interaktivita 49
7.1. Ovládací prvky 50
7.2. Indikační prvky 50

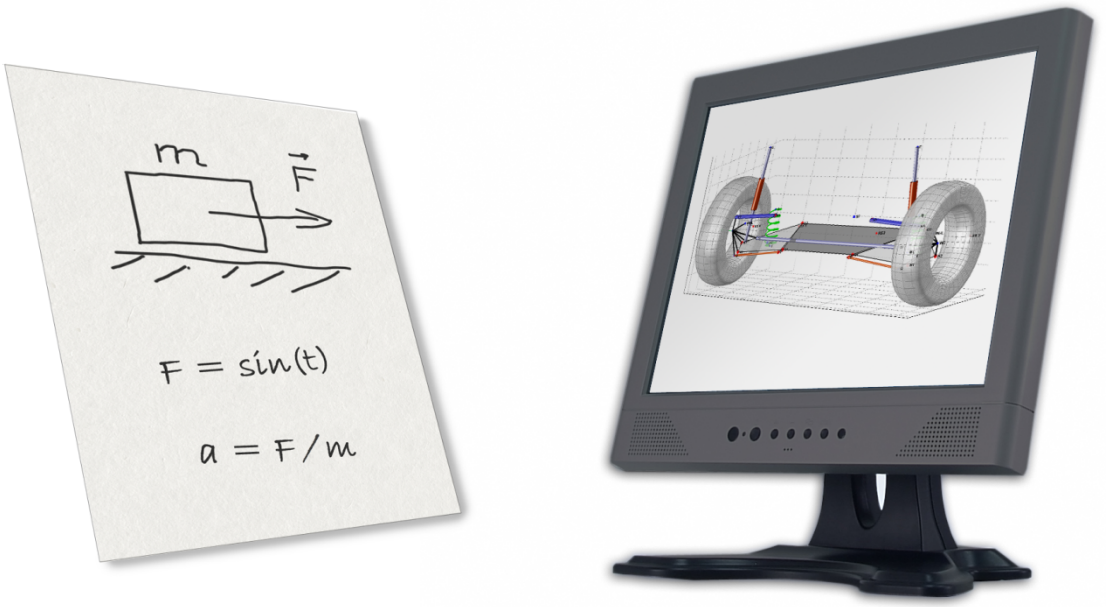
7.3. Další možnosti 51
8. Virtuální realita 52
8.1. Pasivní stereoskopie 52
8.1.1. Výhody 53
8.1.2. Nevýhody 53
8.2. Aktivní stereoskopie 54
8.2.1. Výhody 55
8.2.2. Nevýhody 55
8.3. Anaglyfní stereoskopie 55
8.3.1. Výhody 56
8.3.2. Nevýhody 56
8.4. Prostorové ozvučení 57
9. Verifikační úlohy 58
9.1. Úloha č. 1 – Prutová soustava 58
9.1.1. Zadání 58
9.1.2. Analytické řešení 58
9.1.3. Numerické řešení 59
9.1.4. Srovnání výsledků 59
9.2. Úloha č. 2 – Klikový mechanismus 60
9.2.1. Zadání 60
9.2.2. Analytické řešení 60
9.2.3. Numerické řešení 61
9.2.4. Srovnání výsledků 61
10. Výkonnostní porovnání 63
10.1. Způsob testování 63
10.1.1. Hardwarová konfigurace 63
10.1.2. Softwarová konfigurace 63
10.2. Testované modely 64
10.3. Výsledky základních testů 65
10.4. Přínos paralelizace 66
10.5. Vliv integračního kroku 67

10.6. Porovnání HW platforem 68
10.7. Shrnutí 69
11. Závěr 70
11.1. Dosažené cíle 70
11.2. Výhody a nevýhody 70
11.3. Směr dalšího vývoje 70
Použité symboly a zkratky 72
Literatura 74
Přílohy 75
Příloha č. 1 – hlavní nástroje GUI 75
Příloha č. 2 – seznam příkazů konzole 76
Příloha č. 3 – ukázkový skript konzole 78

1. Úvod

Základním rysem moderní doby je zejména obrovské tempo vývoje společnosti, vědomostí a technologií. Snad nejvíce dnešní dobu ovlivňuje rozmach počítačů, který se započal v druhé polovině 20. století a stále pokračuje. Počítače dnes slouží nejrůznějším účelům a stala se z nich běžná součást života prakticky každého civilizovaného jedince. Posílila se i jejich původní úloha ve vědě a technice, takže dnes je již prakticky nepředstavitelné, že by se alespoň trochu složitější výrobek navrhoval bez pomoci počítačových programů.

S rostoucím výkonem výpočetní techniky se zvyšuje i složitost a schopnosti softwarových prostředků, které jsou užívány již od počátečního návrhu výrobku, přes případnou simulaci funkčnosti až k řízení samotné výroby. Díky tomu se mohou řešit stále složitější úlohy v kratším čase.



Obrázek 1: Pokrok ve výpočetní technice

Oblast aplikací zabývající se podporou navrhování pomocí počítačů se souhrnně nazývá CAD (Computer Aided Design), či CAE (Computer Aided Engineering). Do těchto skupin spadají různé 2D a 3D rýsovací či modelovací systémy (např. AutoCAD, Pro/Engineer, Inventor), výpočetní systémy využívající metody konečných prvků (např. Ansys), simulace proudění tekutin (např. Fluent) a mimo jiné i multi-body systémy. Posledním zmiňovaným zástupcem rodiny CAD/CAE systémů se bude zabývat právě tato diplomová práce.

1.1. MULTI-BODY SYSTÉMY

Stručně řečeno jsou multi-body systémy výpočetní programy sloužící k simulaci pohybu a vzájemného působení soustavy těles, které jsou mezi sebou pospojovány různými kinematickými vazbami či jinými silovými prvky a zatíženy obecně proměnnými vnějšími silovými účinky.

Multi-body systémy mohou sloužit například jako prostředek pro přípravu dat, jež jsou poté využívány v dalších systémech výpočetního procesu, kterým prochází vývoj mnoha různých výrobků. Dalším možným využitím je simulace a ověřování funkčnosti mechanických systémů.

1.1.1. ROZDĚLENÍ MULTI-BODY SYSTÉMŮ

Multi-body systémy lze rozčlenit podle mnoha různých kritérií. Základní skupiny tvoří rozdělení podle:

A) druhu pohybu těles
- **LINEÁRNÍ** – základní těleso koná velký nelineární pohyb, ale závislá tělesa konají vůči základnímu tělesu malé lineární pohyby, což umožňuje zjednodušit a zrychlit celé řešení, ovšem použitelnost je omezena pouze na určité typy mechanismů
- **NELINEÁRNÍ** – všechna tělesa mohou konat velké nelineární pohyby; komplexní, ale pomalejší řešení umožňuje simulaci v zásadě všech typů mechanismů

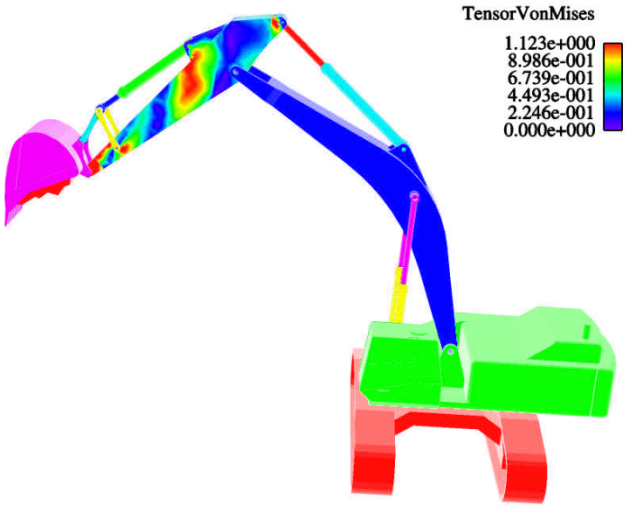
B) formalizmu
- **SYMBOLICKÉ** – sestavení pohybových rovnic v symbolické formě (např. jazyka C nebo FORTRAN) se provede pouze na začátku procesu řešení; poté se provede optimalizace a překlad do strojového jazyka, což za cenu delší přípravné fáze umožňuje teoreticky rychlejší výpočet
- **NUMERICKÉ** – sestavení a výpočet pohybových rovnic probíhá vnitřně v každém integračním kroku

C) implementace
- **PARALELIZOVATELNÉ** – metoda a její implementace použitá k řešení umožňuje rozdělení výpočtu do vícero paralelně vykonávaných procesů, což dovoluje lépe využít výkonu víceprocesorových systémů
- **NEPARALELIZOVATELNÉ** – metoda použitá k řešení rozdělení výpočtu do paralelních procesů neumožňuje, nebo není takto implementována

1.1.2. MODERNÍ TRENDY

Hlavním trendem všech výpočetních aplikací je samozřejmě zvyšování výkonu a schopností daného systému. Moderní aplikace se snaží být zároveň mocné, ale i jednoduché a intuitivní na ovládání, což se ne vždy daří skloubit.

Dalším trendem je postupné slučování funkcí multi-body systému a systému pro výpočet namáhání pomocí MKP. Například multi-body systém ADAMS umožňuje zahrnout do výpočtu mechanismu i pružná tělesa, jejichž deformační mód je předpočítán v MKP softwaru.



Obrázek 2: Pružné těleso v multi-body systému ADAMS (horní rameno jeřábu) [3]

1.2. MULTI-BODY SYSTÉM MECHANICS

Jak již bylo zmíněno výše, tato diplomová práce se zabývá tématem multi-body systémů – konkrétně tvorbou nového multi-body systému pojmenovaného Mechanics.

Na základě předchozího úvodu by se tento multi-body systém dal charakterizovat jako nelineární numerický s podporou paralelizace. Hlavní důraz při vývoji systému Mechanics je kladen na rychlost a interaktivitu simulace.

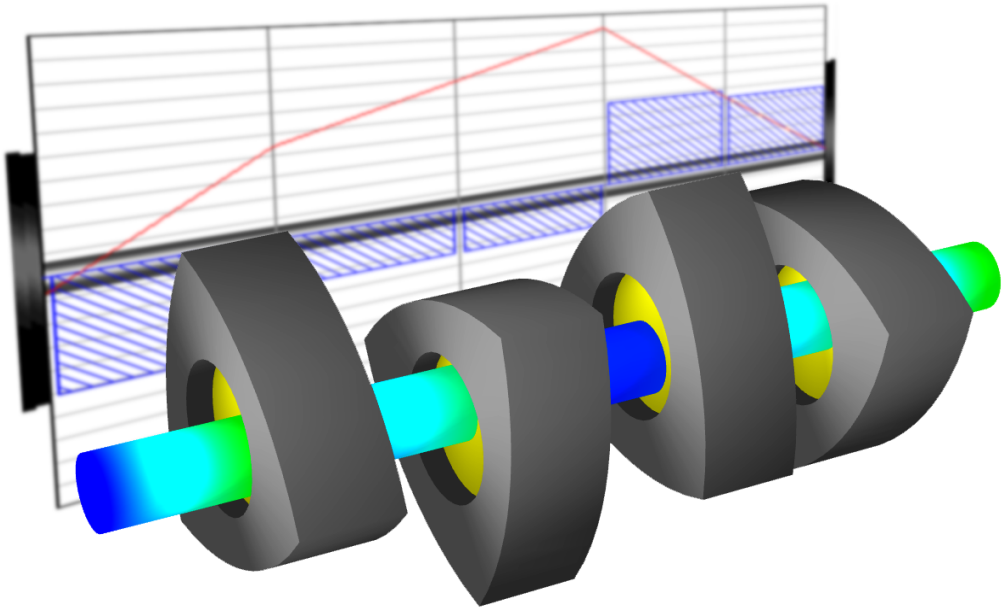
Obrázek 3: Logo aplikace Querté Mechanics

2. METODA KOREKČNÍCH SIL A JEJÍ VYUŽITÍ

2.1. ZÁKLADNÍ CHARAKTERISTIKA METODY

Metoda korekčních sil (dále jen MKS) je původní numerická metoda určená pro souběžné řešení dynamiky a kinematiky jednoduchých i složitých mechanismů pomocí moderní výpočetní techniky. Přestože hlavním cílem MKS je určení reakčních sil kinematických vazeb spojujících dvě či více těles, tak požadavky vyplývající z implementace této metody ovlivňují i další části řešení multi-body modelů a proto bude nadále v textu používáno označení „MKS systém“.

Původním záměrem vývoje MKS byla schopnost simulace torzního kmitání složené vícečlánkové hřídele Wankelova rotačního motoru, jakožto součásti výpočetního a simulačního systému EngSim Next (Obrázek 4). V dalším průběhu vývoje se ovšem tato metoda ukázala býti soběstačným a plnohodnotným prostředkem pro sestavení obecného multi-body systému, kterému navíc propůjčuje řadu zajímavých vlastností.



Obrázek 4: Původní účel MKS v aplikaci EngSim Next

Metodu korekčních sil lze zařadit současně do rodiny jednokrokových i vícekrokových explicitních numerických metod, což je dáno tím, že při řešení jednoho konkrétního mechanismu se mohou uplatňovat oba výše zmíněné typy. Zevrubnějšímu teoretickému popisu použitých algoritmů a matematického aparátu metody je věnována samostatná kapitola.

Mnohé vlastnosti zmiňované dále v textu se úzce vážou na již konkrétní implementaci principů MKS. Proto bude-li se hovořit o MKS, je tím myšlena právě současná podoba implementovaná v aplikaci Mechanics. Týká se to zejména popisovaných datových typů a struktur, nebo možností a postupů spojených s využíváním MKS.

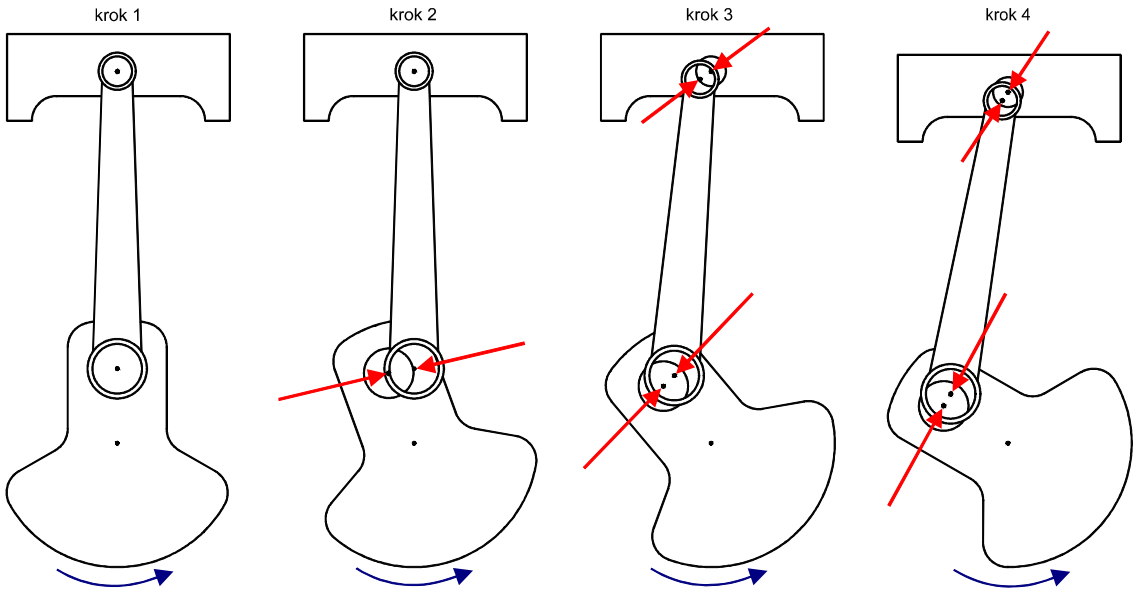
2.2. PRINCIP MKS

Na příkladu klikového mechanismu (Obrázek 5) je zjednodušeně znázorněn princip MKS v průběhu čtyř kroků simulace. Mezi čepem klikového hřídele a dolním okem ojnice je válcová vazba, stejně jako mezi horním okem ojnice a pístním čepem. Kliková hřídel i píst jsou mimo to ještě příslušnými vazbami spoutány s nějakým statickým tělesem, což ovšem není z důvodu přehlednosti v obrázku vyznačeno.

Na začátku každého iteračního kroku simulace se na základě silového působení z minulého kroku simulace vypočítá nový kinematický stav všech těles podle standardních pohybových rovnic Newtonovské mechaniky. Poté se právě prostřednictvím MKS určí reakční síly kinematických vazeb. Tyto reakční síly spolu s definovaným vnějším zatížením opět vytvoří nové silové působení, podle kterého se analogicky v dalším simulačním kroku vypočítají nové kinematické stavy těles. Celý proces se nadále stejným způsobem opakuje.

Pro větší názornost následuje popis průběhu simulace pro konkrétní mechanismus klikového ústrojí popsaného výše. Průběh simulace po jednotlivých iteračních krocích pak probíhá v tomto sledu:

1. Všechna tělesa jsou v klidu, neboť na ně nepůsobí ještě žádné síly či momenty. Na klikovou hřídel nyní začne působit vnější kroutící moment.
2. Kliková hřídel se dá do pohybu, ale ojnice i píst stále stojí. Rozdíl poloh vázaných bodů mezi klikovou hřídelí a ojnicí vyvodí korekční sílu, která se snaží podle zákona akce a reakce vrátit obě vázaná tělesa k sobě.
3. Klikový hřídel se stále pohybuje, ale kromě vnějšího kroutícího momentu na něj již působí i korekční síla mezi ním a ojnicí. Stejná síla opačného směru ovšem způsobí, že se začne pohybovat i ojnice. Píst stále stojí. Rozdíl poloh vázaných bodů mezi ojnicí a pístem vyvodí opět korekční síly mezi těmito dvěma tělesy.
4. V důsledku silového působení se začal pohybovat už i píst. Nyní se pohybují již všechna tělesa. Rozdíl poloh vázaných bodů vyvodí opět korekční síly…



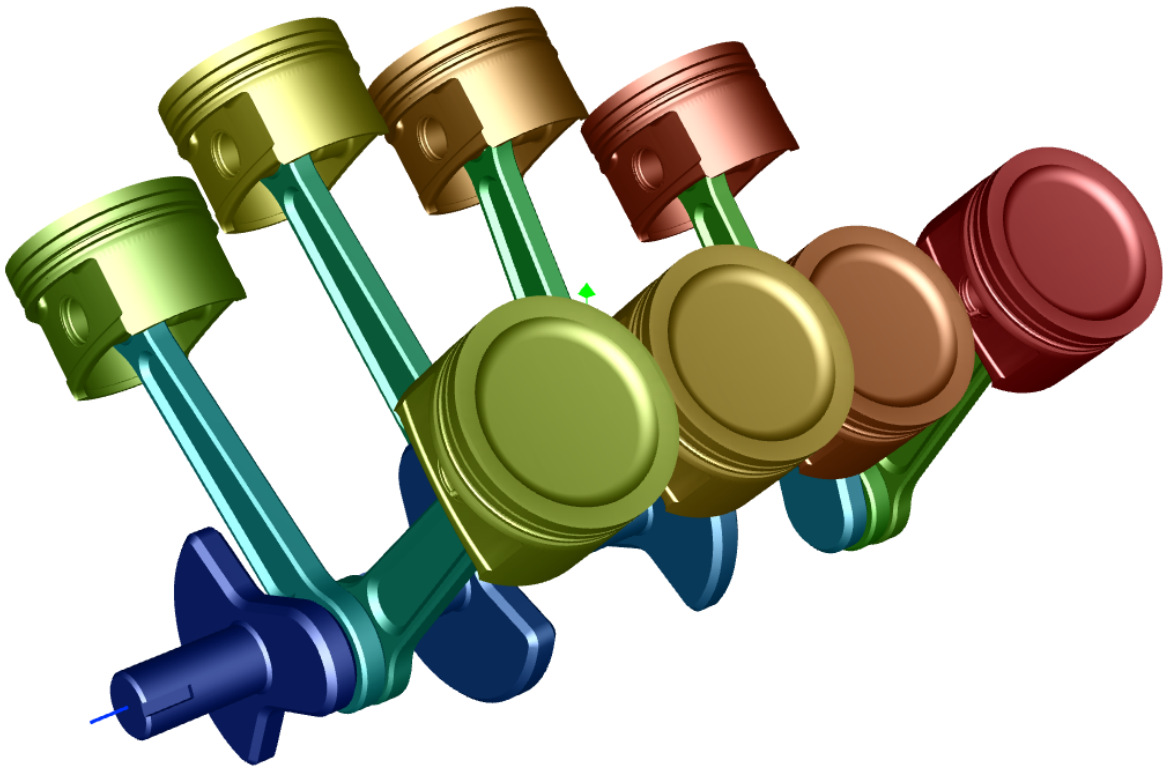
Obrázek 5: Znázornění principu MKS během čtyř iterací (červeně jsou zobrazeny korekční síly vazeb a modře vnější silové působení)

Rozdíly poloh vázaných bodů znázorněné na obrázku jsou ve skutečnosti mnohem menší, neboť i samotný časový diskretizační krok simulace je velice jemný (řádově obvykle kolem mikrosekundy).

3. ELEMENTÁRNÍ PRVKY MKS

3.1. TYPY MECHANISMŮ

Pomocí MKS je možné řešit každý mechanismus v trojrozměrném prostoru, který lze sestavit pomocí základních podporovaných elementů, k nimž patří tuhá tělesa, kinematické vazby, pružné linky a vnější síly. Počet jednotlivých elementů a stupňů volnosti simulovaného mechanismu je prakticky neomezený.



Obrázek 6: Příklad mechanismu vytvořeného v MKS systému Mechanics

3.2. JEDNOTKY

V MKS systému Mechanics se nepoužívají jednotky veličin. Výsledky vycházejí ve stejné jednotkové skupině, v jaké jsou zadány vstupní hodnoty. Standardně se uvažuje výchozí jednotková skupina SI (kilogram, metr, sekunda, Newton, atd.).

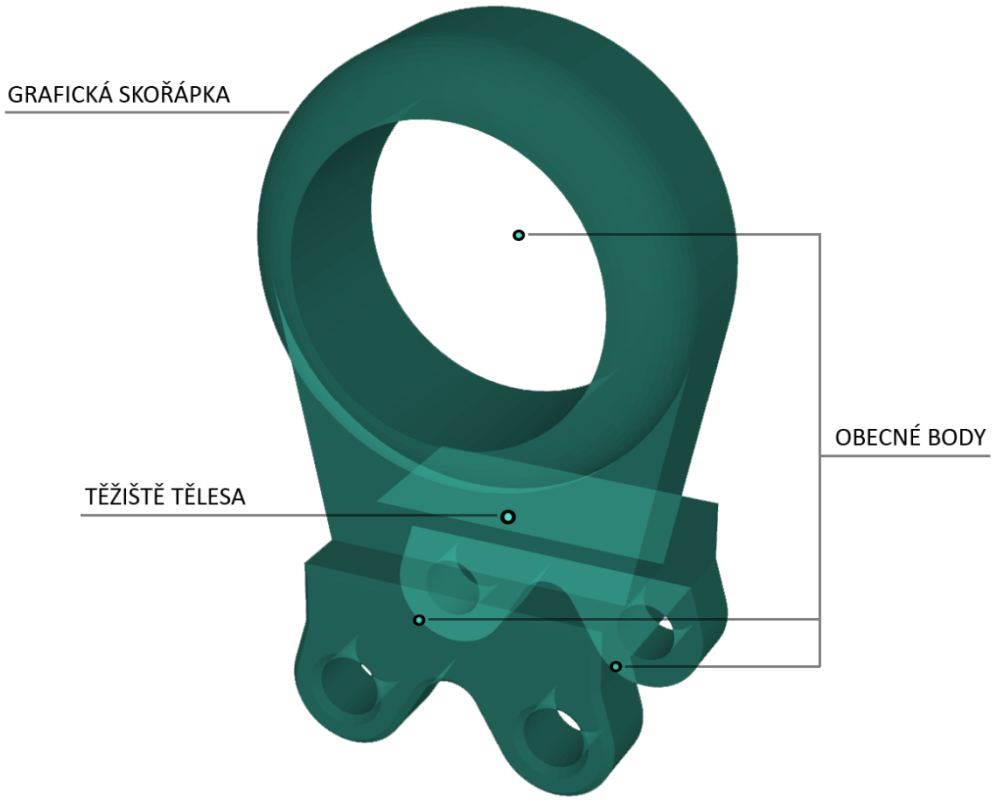
3.3. ZÁKLADNÍ ELEMENTY

V následujících odstavcích budou stručně popsány jednotlivé základní elementy, z kterých je možno sestavovat funkční modely. Celý systém Mechanics využívá pojmenování prvků v angličtině, proto jsou v závorkách za názvy prvků uváděny i jejich anglické ekvivalenty.

3.3.1. TĚLESA (BODIES)

Tuhé těleso je základní stavební prvek každého mechanismu. Těleso může obsahovat jeden či více bodů, ke kterým mohou být využitím prvků typu vazba nebo link připojena další tělesa. Také je možno těchto bodů využívat jako kinematických či dynamických senzorů.

První bod tělesa se od zbývajících bodů funkčně mírně liší, neboť definuje jeho těžiště. K těžišti tělesa se vztahuje i absolutní poloha a natočení tělesa v globálním souřadném systému. Ostatní body tělesa jsou definovány vektory relativního posunu a natočení právě vůči těžišti.



Obrázek 7: Znázornění elementu tělesa

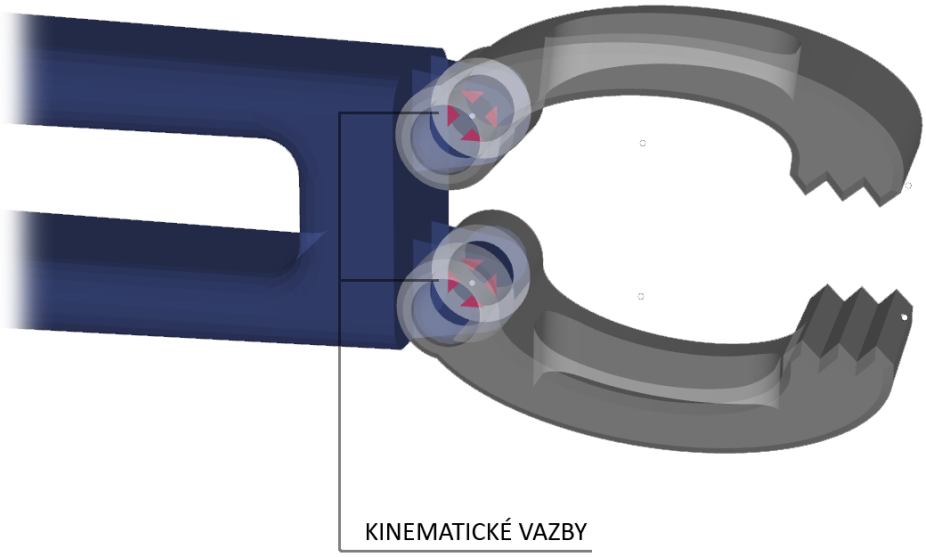
Kromě jednoho či více bodů musí mít každé těleso definovanou také hmotnost a tenzor setrvačnosti. Multi-body systém Mechanics umožňuje existenci i nekonečně hmotných těles, u nichž pak žádné vnější silové působení neovlivňuje jejich pohybový stav (který lze ovšem stále ovládat přímým zápisem hodnot do kinematických vektorů).

3.3.2. VAZBY (BINDS)

Kinematická vazba spojuje právě dva body různých těles a předepisuje jim různé možnosti vzájemného pohybu ve zvoleném vztažném souřadném systému v závislosti na nastavení vazebných koeficientů.

Vazby k obecným matematicky popsatelným plochám nebo křivkám metoda korekčních sil principiálně umožňuje, ovšem současná implementace tuto možnost nepodporuje.

V některých multi-body systémech je nutno rozlišovat mezi takzvanými otevřenými mechanismy (tělesa netvoří uzavřený řetězec a mechanismus obsahuje volné konce) a uzavřenými mechanismy (opak předchozího případu). MKS ovšem z principu mezi těmito typy mechanismů nerozlišuje a je tudíž schopna řešit bez rozdílu v možnostech nebo výkonnosti oba typy.



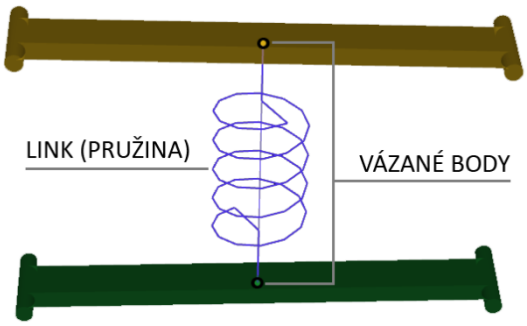
Obrázek 8: Znázornění kinematických vazeb

3.3.3 LINKY (LINKS)

Link je koncipován jako multifunkční spojovací prvek, který váže obdobně jako prvek vazby dva body různých těles. V současné implementaci umožňuje následující čtyři základní typy chování:

- Pružina
- Progresivní pružina
- Torzní pružina
- Torzní progresivní pružina

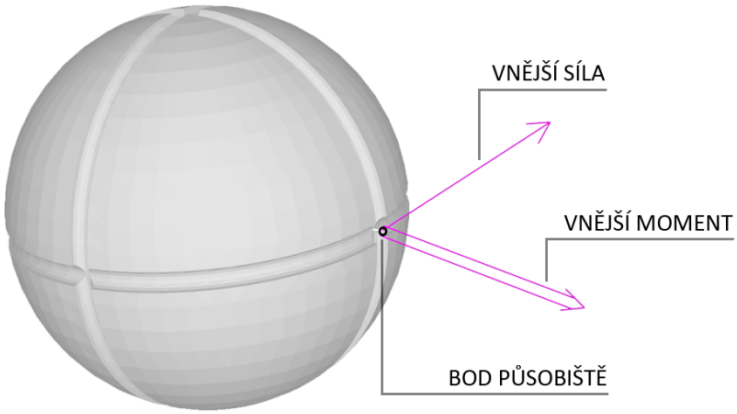
Jednotlivé typy lze dále modifikovat nastavení koeficientů pro přenos tlakových a tahových sil. U torzních linků je možné nastavit také převodový poměr, čímž lze simulovat jednoduché rotační převody.



Obrázek 9: Znázornění linku

3.3.4. Síly (Forces)

Vnější síly jsou prvky modelu, které se vážou na jeden bod tělesa, ve kterém během simulace vytvářejí vnější silové a momentové působení. Element je pak definován vektorem natočení souřadného systému a vektory udávajícími velikost síly a momentu působících na zatěžovaný bod tělesa (Obrázek 10).



Obrázek 10: Znázornění elementu vnější síly (silová složka je znázorněna jednoduchou šipkou, zatímco momentová složka šipkou dvojitou)

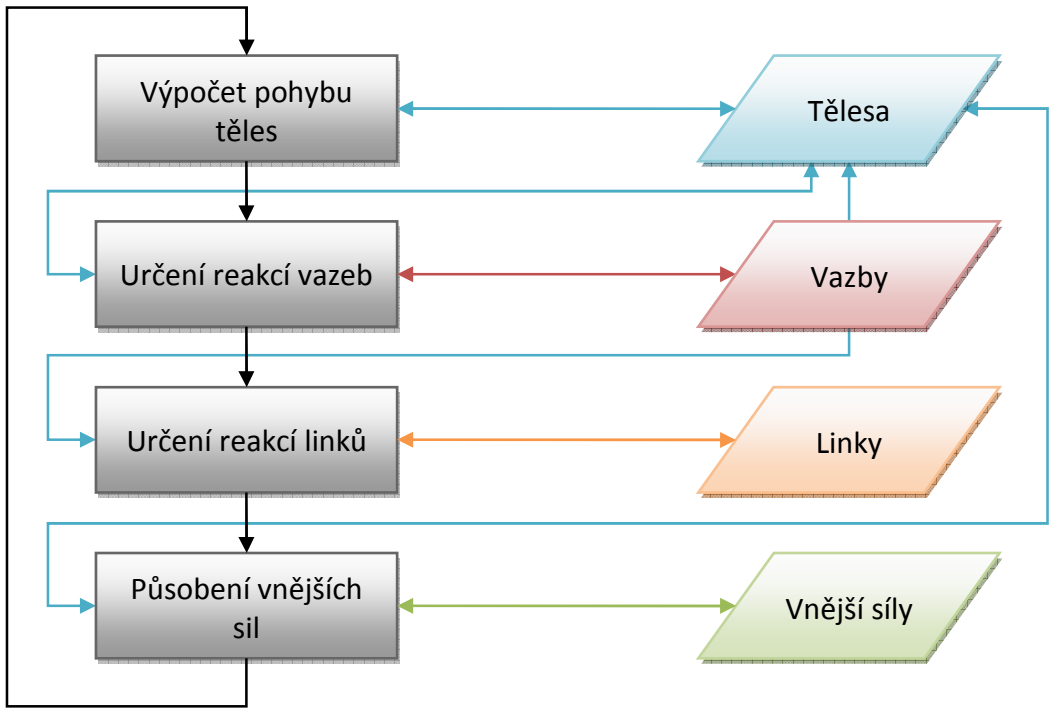
Speciálním případem vnějšího silového působení v modelu je gravitace, která působí definovaným vektorem síly na všechna tělesa v bodě jejich těžiště.

4. POSTUP ŘEŠENÍ

4.1. VÝPOČETNÍ SCHÉMA

Na následujícím schématu (Obrázek 11) je symbolicky znázorněn průběh jedné iterace výpočetního postupu MKS. Šedé bloky symbolizují výpočetní rutiny a barevné bloky pak data jednotlivých elementů.

Ze schématu je vidět, že při výpočtu všech typů elementů se přistupuje k datům těles, což je důležitý fakt z hlediska paralelizace a ošetření paměťových kolizí, což bude podrobněji zmíněno v kapitole věnující se programové implementaci MKS v systému Mechanics.



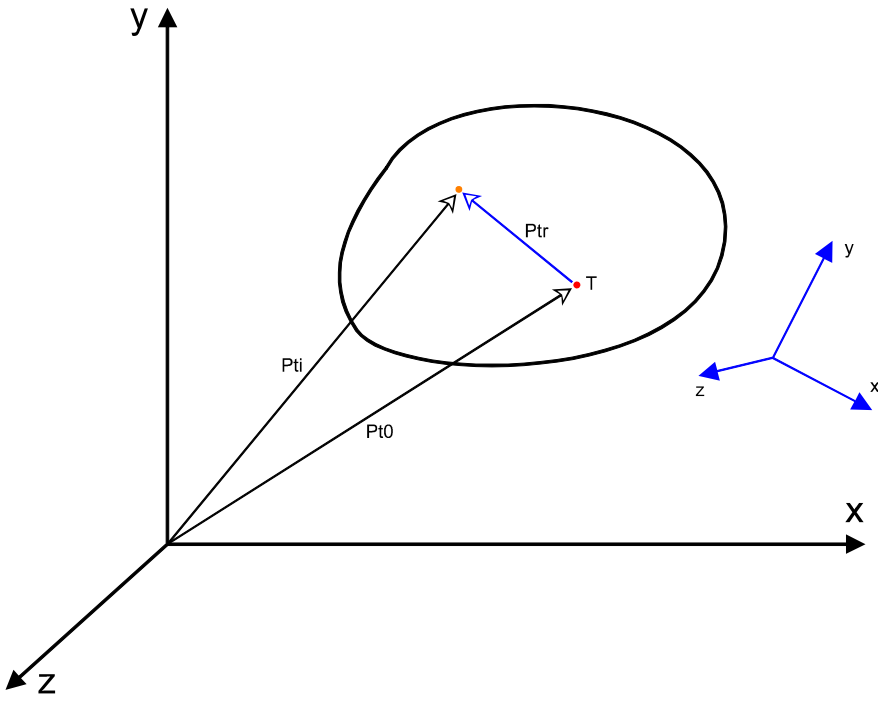
Obrázek 11: Blokové schéma výpočtu

Kromě kroků uvedených výše se při každé iteraci řeší ještě další programové bloky související s řízením jádra, prováděním rozšiřujících modulů, pořizováním záznamů simulovaných veličin a s vykreslovacími rutinami. Tyto výpočty ovšem souvisí přímo s implementací systému Mechanics a pro teoretický rozbor samotného fungování MKS nejsou podstatné. Následující rozbor se bude tedy týkat pouze čtyř základních kroků uvedených na schématu.

4.2. VÝPOČET POHYBU TĚLES

Pozici každého tělesa v prostoru GSS definují dva vektory, z nichž první určuje posunutí LSS těžiště tělesa vůči počátku GSS a druhý pak natočení tohoto LSS opět vůči osám GSS. Pro popis natočení je používán systém postupných rotací kolem os X, Y a Z. Pohyb tělesa je pak v každé iteraci určován na základě Newton-Eulerových pohybových rovnic. Kompletní výpočet elementu tělesa pak probíhá následovně:

- Určení silových a momentových výslednic k těžišti tělesa
- Aktualizace kinematického stavu těžiště tělesa
- Aktualizace kinematického stavu dalších bodů tělesa



Obrázek 12: Těleso (modře je znázorněno natočení lokálního souřadného systému bodu těžiště)

4.2.1. URČENÍ SILOVÝCH A MOMENTOVÝCH VÝSLEDNIC K TĚŽIŠTI TĚLESA

Body tělesa (Obrázek 12) jsou indexovány od nuly až do n-1, kde n je počet bodů. První bod (tj. bod s indexem nula) je těžiště tělesa.

$$\vec{F}_{cg} = \sum_{i=0}^{n-1} \vec{F}_i + m \cdot \vec{g} \qquad (1)$$

$$\vec{M}_{cg} = \sum_{i=0}^{n-1} (\vec{M}_i + \vec{P}_{tr} \times \vec{F}_{i_i}) \qquad (2)$$

Po určení silových a momentových výslednic v těžišti tělesa se všem bodům (včetně těžiště) vynulují vektory výslednic, neboť v dalším iteračním kroku se k nim budou opět od začátku přičítat všechny síly a momenty od vazebných prvků či vnějších působitelů.

$$\forall i \in \langle 0 | n-1 \rangle : \tag{3}$$

$$\vec{F}_i = \vec{0} \tag{4}$$

$$\vec{M}_i = \vec{0} \tag{5}$$

4.2.2. Aktualizace kinematického stavu těžiště tělesa

Kinematický stav tělesa, resp. jeho těžiště, se určuje na základě Newton-Eulerových pohybových rovnic. Translační složku pohybu popisují klasické Newtonovy pohybové rovnice (Rovnice 6, 7, 8).

$$\vec{A}_t = \frac{\vec{F}_{cg}}{m} \tag{6}$$

$$\vec{V}_t = \int \vec{A}_t \cdot dt \tag{7}$$

$$\vec{P}_t = \int \vec{V}_t \cdot dt \tag{8}$$

Rotační složku pohybu pak popisují následující tři rovnice. Rotační zrychlení je určeno na základě Eulerovy pohybové rovnice (Rovnice 9) a prostou integrací se vypočítá i okamžitá rotační rychlost (Rovnice 10). Vztah pro výpočet rotační polohy (Rovnice 11) je převzat z literatury [1].

$$\vec{A}_r = I^{-1} \cdot (\vec{M}_{cg} - \vec{V}_r \times I \cdot \vec{V}_r) \tag{9}$$

$$\vec{V}_r = \int \vec{A}_r \cdot dt \tag{10}$$

$$\vec{P}_r = \int (Q^{-1} \cdot \vec{V}_r) \cdot dt \tag{11}$$

4.2.3. Aktualizace kinematického stavu dalších bodů

Poté, co je vypočítán kinematický stav bodu těžiště, určí se již snadno i kinematické stavy zbývajících bodů tělesa. Translační poloha bodu se získá přičtením relativního vektoru posunutí k translační poloze těžiště. Vektor rychlosti resp. zrychlení se

vypočítá jako derivace polohy resp. rychlosti. Rotační zrychlení, rychlost i poloha bodů je shodná s těžištěm tělesa.

$$\forall i \in \langle 1 | n-1 \rangle : \tag{12}$$

$$\vec{P}_{t_i} = \vec{P}_{t_0} + C^T(\vec{P}_{r_0}) \cdot \vec{P}_{tr_i} \tag{13}$$

$$\vec{P}_{r_i} = \vec{P}_{r_0} \tag{14}$$

$$\vec{V}_{t_i} = \frac{\Delta \vec{P}_{t_i}}{\Delta t} \tag{15}$$

$$\vec{V}_{r_i} = \vec{V}_{r_0} \tag{16}$$

$$\vec{A}_{t_i} = \frac{\Delta \vec{V}_{t_i}}{\Delta t} \tag{17}$$

$$\vec{A}_{r_i} = \vec{A}_{r_0} \tag{18}$$

4.3. Výpočet korekčních sil vazeb

Výpočet korekční síly a momentu vazby probíhá následovně:

- Určení diferenčních vektorů kinematických veličin vázaných bodů
- Výpočet korekčních sil
- Aplikace výsledných sil do vázaných bodů

4.3.1. Určení diferenčních vektorů kinematických veličin vázaných bodů

Prvním krokem při výpočtu vazeb je určení diferenčních vektorů translační i rotační polohy, rychlosti a zrychlení mezi prvním a druhým vázaným bodem v globálním souřadném systému. Tyto diferenční vektory se násobením transformační maticí převedou do lokálního souřadného systému vazby.

$$\vec{A}_{t_d} = C \cdot (\vec{A}_{t_A} - \vec{A}_{t_B}) \tag{19}$$

$$\vec{A}_{r_d} = C \cdot (\vec{A}_{r_A} - \vec{A}_{r_B}) \tag{20}$$

$$\vec{V}_{t_d} = C \cdot (\vec{V}_{t_A} - \vec{V}_{t_B}) \tag{21}$$

$$\vec{V}_{r_d} = C \cdot (\vec{V}_{r_A} - \vec{V}_{r_B}) \tag{22}$$

$$\vec{P}_{t_d} = C \cdot (\vec{P}_{t_A} - \vec{P}_{t_B}) \tag{23}$$

$$\vec{P}_{r_d} = C \cdot (\vec{P}_{r_A} - \vec{P}_{r_B}) \tag{24}$$

Výpočet diferenčního vektoru natočení (Rovnice 24) lze v uvedené formě použít jen pro malé úhlové rozdíly. V obecné formě by bylo nutné použití dodatečných transformací, nicméně pro výpočet vazeb je zmíněné řešení dostačující.

4.3.2. Výpočet korekčních sil

Dalším krokem je určení vektorů korekční síly a momentu. Součiny mezi vektory v následujících rovnicích jsou prováděny po složkách. Tato operace bývá též označována jako Hadamardův součin.

$$\vec{F}_K = \vec{F}_{K_{k-1}} \bullet \vec{K}_{a_{S1}} + \vec{A}_{t_d} \bullet \vec{K}_{t_{S1}} + \vec{V}_{t_d} \bullet \vec{K}_{t_{S2}} + \vec{P}_{t_d} \bullet \vec{K}_{t_{S3}} \tag{25}$$

$$\vec{M}_K = \vec{M}_{K_{k-1}} \bullet \vec{K}_{a_{S2}} + \vec{A}_{r_d} \bullet \vec{K}_{r_{S1}} + \vec{V}_{r_d} \bullet \vec{K}_{r_{S2}} + \vec{P}_{r_d} \bullet \vec{K}_{r_{S3}} \tag{26}$$

4.3.3. Aplikace výsledných sil do vázaných bodů

Posledním krokem je zpětné převedení výsledných vektorů korekční síly a momentu do globálního souřadného systému násobením transponovanou transformační maticí. Poté se již transformované vektory přičtou k silovým výslednicím vázaných bodů.

$$\vec{F}_{P_A} = \vec{F}_{P_A} + C^T \cdot \vec{F}_K \tag{27}$$

$$\vec{F}_{P_B} = \vec{F}_{P_B} - C^T \cdot \vec{F}_K \tag{28}$$

$$\vec{M}_{P_A} = \vec{M}_{P_A} + C^T \cdot \vec{M}_K \tag{29}$$

$$\vec{M}_{P_B} = \vec{M}_{P_B} - C^T \cdot \vec{M}_K \tag{30}$$

4.4. VÝPOČET REAKČNÍCH SIL LINKŮ

Jak již bylo zmíněno v kapitole 3.3.3., elementu typu link lze nastavit různé typy chování (lineární pružina, torzní pružina, atd.). Pro znázornění postupu následuje popis výpočtu linku, jehož mód odpovídá klasické lineární pružině s tlumičem. Výpočet zbývajících módů se liší v dílčích krocích, ovšem základní princip je zachován.

Výchozí krok při výpočtu reakčních sil pružiny s tlumičem je určení délky spojnice vázaných bodů (Rovnice 31) a jednotkového vektoru jejího směru (Rovnice 32). Z této délky a její první derivace podle času se následně vypočítá velikost reakční síly pružiny, přičemž se zde uplatňují koeficienty tuhosti a tlumení (Rovnice 33).

$$L = \left| \vec{P}_{t_B} - \vec{P}_{t_A} \right| \tag{31}$$

$$\vec{e} = \frac{\vec{P}_{t_B} - \vec{P}_{t_A}}{L} \tag{32}$$

$$F = h \cdot L + d \cdot \frac{dL}{dt} \tag{33}$$

Po výpočtu velikosti reakční síly se tato ještě zkoriguje koeficienty linku pro přenos tlakových či tahových sil (Rovnice 34). Touto úpravou lze docílit toho, že se link bude chovat například jako lano, tj. nebude přenášet žádné tlakové reakční síly mezi vázanými body.

$$\begin{aligned} F > 0 &\Rightarrow F = F \cdot u \\ F < 0 &\Rightarrow F = F \cdot v \end{aligned} \tag{34}$$

Na závěr se obdobně jako u reakcí vazeb provede aplikace výsledné reakční síly linku do vázaných bodů (Rovnice 35, 36). Vypočítaná a zkorigovaná velikost reakční síly linku se tedy vynásobí jednotkovým vektorem spojnice vázaných bodů a výsledek se k těmto bodům přičte, resp. odečte.

$$\vec{F}_{P_A} = \vec{F}_{P_A} + \vec{e} \cdot F \tag{35}$$

$$\vec{F}_{P_B} = \vec{F}_{P_B} - \vec{e} \cdot F \tag{36}$$

4.5. VÝPOČET VNĚJŠÍHO SILOVÉHO PŮSOBENÍ

Posledním krokem ve výpočtu jedné iterace algoritmu MKS je aplikace působení od vnějších sil a momentů. Po transformaci zadaných vektorů vnější síly a momentu do zvoleného lokálního souřadného systému se tyto přičtou k silovým výslednicím bodu působiště (Rovnice 37, 38).

$$\vec{F}_P = \vec{F}_P + C \cdot \vec{F} \tag{37}$$

$$\vec{M}_P = \vec{M}_P + C \cdot \vec{M} \tag{38}$$

5. OPTIMALIZACE VAZEBNÝCH PARAMETRŮ

5.1. STABILITA SIMULACE

Přestože je metoda korekčních sil z principu velice stabilní, může v některých případech dojít k tomu, že řešení validního mechanismu zkolabuje, nebo se příliš sníží jeho přesnost.

Numerické výpočty simulace mohou vlivem omezené přesnosti počítačů při výpočtech s čísly s plovoucí desetinnou čárkou anebo vlivem rozkmitání netlumených vazeb začít divergovat. K tomuto druhu nestability dochází pouze ojediněle.

Častější případ nestability vzniká při vysokých rychlostech vázaných těles, kdy změny silového působení na tělesa jsou rychlejší, než je schopna vazba se svou reakční dobou pokrýt. Korekční síly pak již nejsou s to udržet vázaná tělesa dostatečně pohromadě, mechanismus se částečně nebo úplně „rozpadne“ a vzájemný pohyb přestane odpovídat zadaným podmínkám.

Řešení spočívá buď ve snížení pracovní rychlosti mechanismu, zkrácení časového kroku simulace, nebo úpravě parametrů vazby. Poslední uvedená možnost bude rozebrána v následujících odstavcích.

5.2. POPIS PARAMETRŮ

Numerické chování vazeb závisí na třech hlavních skupinách parametrů. Jsou to matice translačních koeficientů K_t, matice rotačních koeficientů K_r a matice doplňkových koeficientů K_a. Matice K_t a K_r se vážou k diferenčním vektorům kinematických veličin vázaných bodů. Matice K_a, resp. její první dva sloupce určují míru přenosu korekčních sil z předchozího kroku řešení do aktuálního. Poslední sloupec matice K_a je nevyužitý.

Poznámka: Uvedené matice nemají pravý matematický maticový význam, ale slouží jako logicky strukturovaný kontejner koeficientů. Sloupce reprezentují koeficientový vektor a nenulové řádky pak odpovídají odebraným stupňům volnosti ve vztahu k uvedeným osám lokálního souřadného systému.

Z rovnic korekčních sil vyplývá, že odebírané stupně volnosti se řídí pouze složkami matic K_t a K_r. Nicméně pro osy, kterým vazba stupně volnosti neodebírá, je vhodné nastavit nulové složky i sloupcovým vektorům přenosu matice K_a. Vlivem numerických nepřesností při transformacích mezi různými souřadnými systémy může totiž docházet k „prosakování“ složek vektorů, tudíž by se mohla jistá nenulová korekční síla objevit i tam, kde by teoreticky být neměla.

Jak již bylo zmíněno, složky koeficientových matic jsou ve vztahu k osám zvoleného lokálního souřadného systému vazby. Tento lokální souřadný systém je určen pouze jedním vektorem, který definuje jeho natočení vůči globálnímu souřadnému systému. Z rovnic korekčních sil vyplývá, že se pracuje s diferenčními vektory, tudíž poloha zdrojových vektorů v lokálním souřadném systému není podstatná.

Kt	Vektor Kt_{S1}	Vektor Kt_{S2}	Vektor Kt_{S3}
Osa X	0,01	1000	10
Osa Y	0,01	1000	10
Osa Z	0	0	0

Kr	Vektor Kr_{S1}	Vektor Kr_{S2}	Vektor Kr_{S3}
Osa X	0	0	0
Osa Y	0	0	0
Osa Z	0,01	1000	10

Ka	Vektor Ka_{S1}	Vektor Ka_{S2}	Vektor Ka_{S3}
Osa X	1	0	0
Osa Y	1	0	0
Osa Z	0	1	0

Tabulka 1: Příklad nastavení parametrů vazby zabraňující vzájemnému pohybu vázaných bodů v osách X a Y a rotaci kolem osy Z

5.3. Vliv přenosu na typ vazby

Vektory Ka_{S1} a Ka_{S2} určují míru přenosu výsledku z předchozího kroku simulace do nového kroku. Významně ovlivňují stabilitu a autokorektivnost vazby.

Pokud je přenos roven nule (resp. složky vektoru), jedná se o pružnou vazbu, neboť reakční působení vazby je ekvivalentní pružině s tlumičem (a nepovinným akceleračním vlivem) natažené mezi vázanými body. Důsledkem toho taková vazba kmitá vlastní frekvencí a vlivem tlumení odebírá část energie systému. Pokud jsou parametry v maticích K_t a K_r nastaveny vhodně, nemusí to být problém a tato vazba má pak některé výhodné vlastnosti.

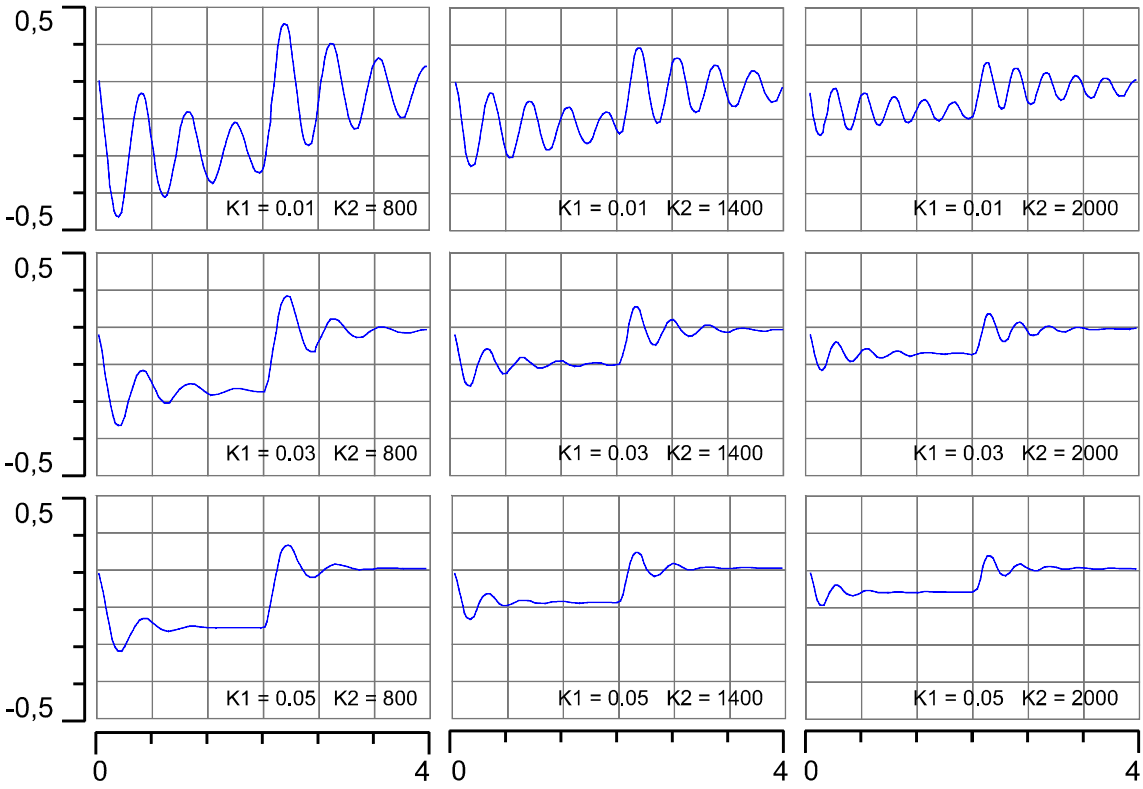
Je-li přenos roven jedné (resp. složky vektoru), jedná se o tvrdou vazbu. Vazba se už nechová přímo jako pružina (spíše by se její chování dalo připodobnit k „časově progresivní pružině“, či k PID regulátoru), tudíž i sklon ke kmitání je menší (a jak se ukáže níže, dá se prakticky eliminovat) a dochází i k menšímu odběru energie ze

systému. Dojde-li důsledkem přetížení vazby k velkému oddálení vázaných bodů, může se vazba stát nestabilní.

5.4. VLIV PARAMETRŮ TVRDÉ VAZBY

Tvrdá vazba s přenosem rovným jedné je nejčastější vazbou, využitelnou spolehlivě ve většině situací.

Následující grafy (Obrázek 13) prezentují vliv různých hodnot koeficientů matice na chování vazby po skokové změně zatížení jednoho z vázaných těles. Ukázkový mechanismus představují dvě tělesa o hmotnosti 1 kg vázané v těžišti. Na první těleso působí střídavě síla ± 10 kN. Pro zjednodušení se uvažuje pouze pohyb v ose X a koeficienty v jednotlivých sloupcích byly nazvány K_1 (pro rozdíly zrychlení), K_2 (rozdíl rychlostí) a K_3 (rozdíl polohy). Na vodorovné ose je vynesen čas (0 až 4 ms) a na svislé ose rozdíl polohy vázaných bodů (-0,5 až 0,5 mm).

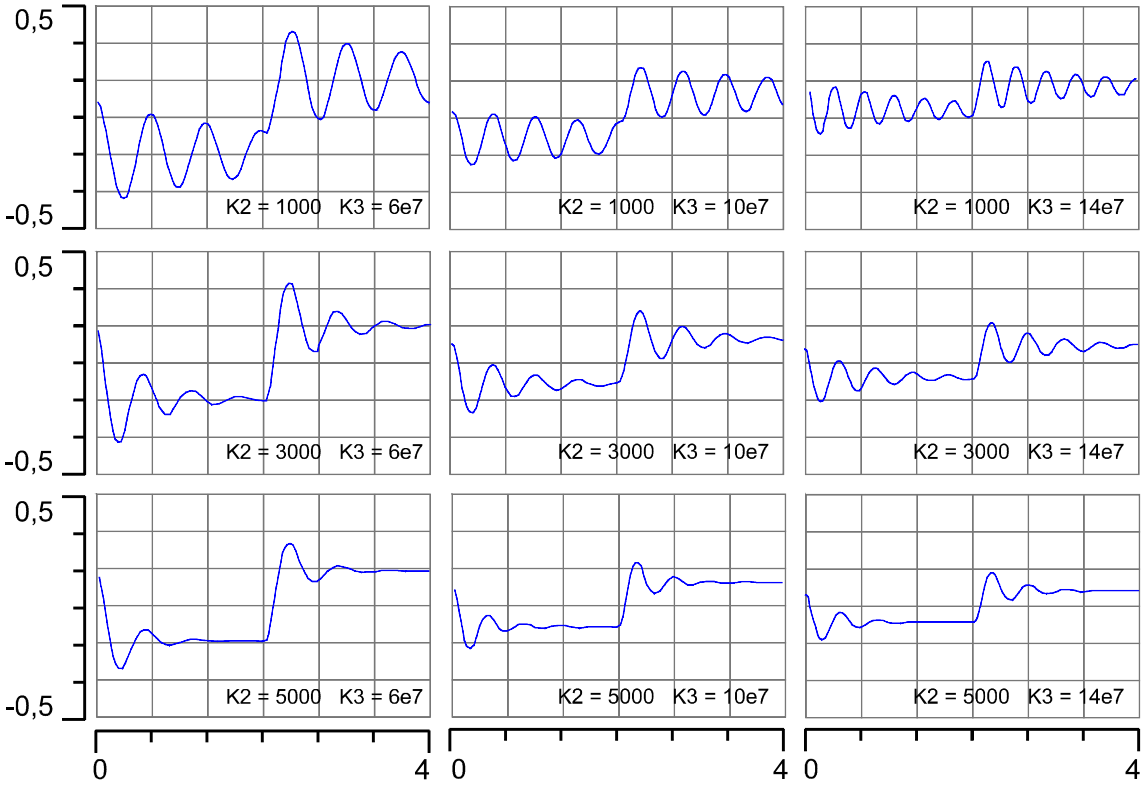


Obrázek 13: Vliv parametrů tvrdé vazby na odezvu při skokové změně zatížení

Z grafů je vidět, že koeficient K_1 má vliv na útlum kmitání, zatímco koeficient K_2 má vliv na střední odchylku. Koeficient K_3 není do grafů zahrnut, neboť se projevuje až při velkém oddálení vázaných bodů, resp. dlouhodobě přidržuje body u sebe, aby se vlivem numerických nepřesností nerozcházely.

5.5. VLIV PARAMETRŮ PRUŽNÉ VAZBY

Pružná vazba s přenosem rovným nule se využívá tehdy, je-li zapotřebí, aby se spojení chovalo jako pružina (např. uchycení tělesa v silentbloku). Dá se jí také využít jako doplňkové složky reprezentující tlumení vazebného mechanismu (např. útlum pohybu u tvrdé posuvné vazby).

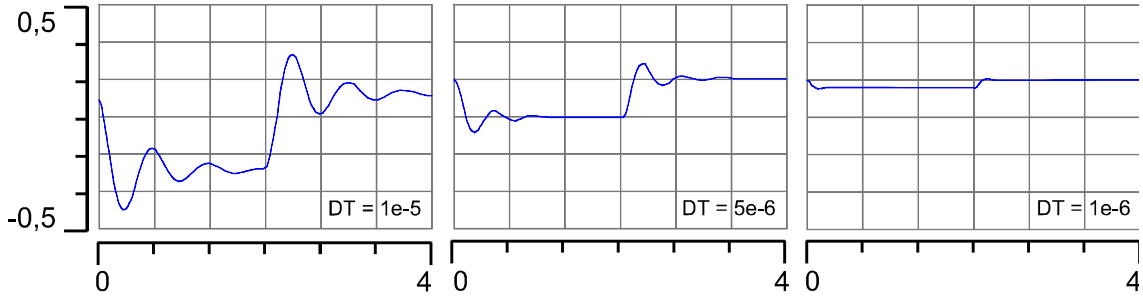


Obrázek 14: Vliv parametrů pružné vazby na odezvu při skokové změně zatížení

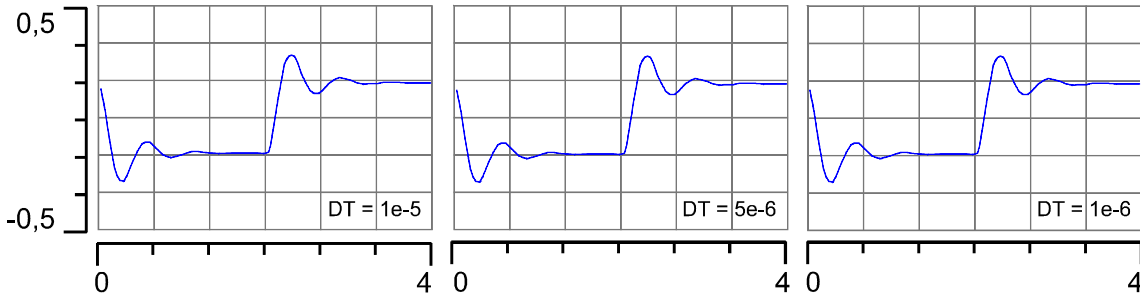
Grafy měření (Obrázek 14) ilustrují, že tlumení tentokrát ovlivňuje koeficient K_2, zatímco koeficient K_3 má vliv na střední odchylku. Koeficient K_1 nebyl do měření zahrnut (byl roven nule), neboť způsoboval divergentní chování vazby. Průběh korekční síly je zdánlivě podobný jako u tvrdé vazby, problém je ovšem v tom, že koeficient K_3 (tuhost vazebné pružiny) je relativně vysoké číslo, což způsobuje potíže při numerických výpočtech. Tento typ vazby je také náchylnější k problémům s rezonančními frekvencemi.

5.6. VLIV ČASOVÉHO KROKU NA VAZBY

Reakce vazby nezávisí pouze na jejich koeficientech, ale také na časovém kroku simulace. Ten ovlivňuje jak průběh reakčních sil, tak i reakční dobu, což je podstatné pro rychlé mechanismy. Způsob vlivu na tvrdou i pružnou vazbu se výrazně liší.



Obrázek 15: Vliv časového kroku na odezvu tvrdé vazby



Obrázek 16: Vliv časového kroku na odezvu pružné vazby

Jak je vidět z grafů, u tvrdé vazby (Obrázek 15) kratší časový krok zmenšuje reakční dobu, zatímco u pružné vazby (Obrázek 16) nemá časový krok vůbec žádný vliv, což je logické a vyplývá to z toho, že se pružná vazba chová jako pružina kmitající na vlastní frekvenci.

Rozumné rozmezí nastavení časového diskretizačního kroku pro typické mechanismy se pohybuje v řádovém pásmu 10^{-5} až 10^{-7}. Čím větší hodnoty (rozuměj delší časový krok), tím je simulace rychlejší, ale snižuje se přesnost řešení. Hodnoty většího řádu než 10^{-5} však mohou způsobovat nestabilitu simulace (záleží ovšem také na rychlostech a hmotnostech vázaných těles).

5.7. Zhodnocení tvrdé a pružné vazby

Výhodou tvrdé vazby je jednoznačně výborná schopnost reakce na skokové změny zatížení, navíc zlepšující se s kratším časovým krokem simulace. To se projevuje příznivě na nízkém odběru energie systému i při vysokých rychlostech. Nevýhodou je nestabilní a divergentní chování po přetížení vazby.

Pružná vazba naproti tomu regeneraci z přetížení zvládá dobře, ovšem zvláště při vysokých rychlostech se může negativně projevovat její tlumící efekt, nebo výrazné rezonanční pásma. Její chování nezávisí na velikosti časového kroku. Výhodná může být pro pomalé a volné mnohočlánkové mechanismy (řetěz), nebo pro přeurčené uložení (dveře na třech pantech), kde naopak tvrdá vazba může způsobovat jisté problémy.

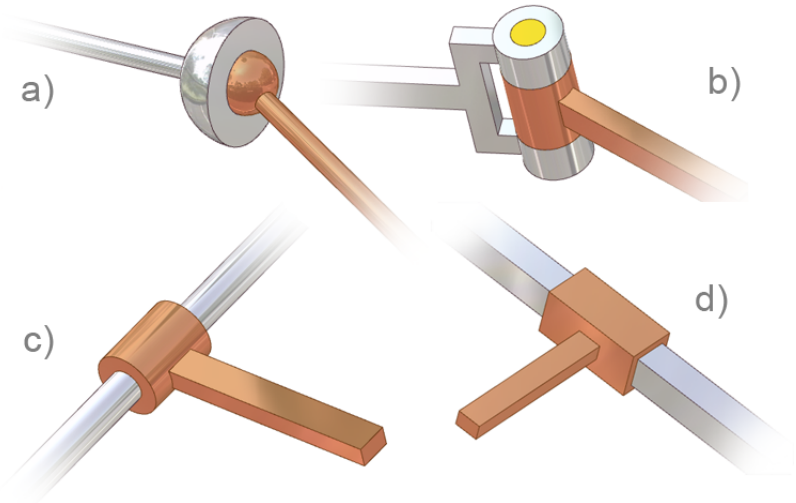
5.8. PŘEDDEFINOVANÉ VAZBY

Zadávání všech vazebných koeficientů pro každou vazbu by bylo jednak zdlouhavé, ale hlavně pro nezkušeného uživatele velmi obtížné. Proto obsahuje pro zjednodušení práce systém Mechanics několik základních předdefinovaných vazeb, u kterých se vazebné koeficienty určí automaticky.

5.8.1. TYPY PŘEDDEFINOVANÝCH VAZEB

Skupina předdefinovaných vazeb v systému Mechanics se snaží pokrýt nejčastější kombinace kinematických vazeb, které se u většiny mechanismů vyskytují. V základní nabídce programu jsou tedy vazby následujícího typu:

- Uživatelská
  - Bez výpočtu momentových reakcí
  - S výpočtem momentových reakcí
- Sférická
- Rotační
  - Kolem osy X
  - Kolem osy Y
  - Kolem osy Z
- Rotační s převodem
  - Kolem osy X
  - Kolem osy Y
  - Kolem osy Z
- Válcová
  - Kolem osy X
  - Kolem osy Y
  - Kolem osy Z
- Válcová s převodem
  - Kolem osy X
  - Kolem osy Y
  - Kolem osy Z
- Posuvná
  - Podél osy X
  - Podél osy Y
  - Podél osy Z
- Plošná
  - V rovině XY
  - V rovině YZ
  - V rovině ZX



Obrázek 17: Znázornění vazeb: a) sférická, b) rotační, c) válcová, d) posuvná

5.8.2. AUTOMATICKÝ ODHAD PARAMETRŮ

Pro všechny předdefinované vazby (vyjma uživatelských) se složky matic vazebných koeficientů vypočítají automaticky podle následujícího postupu:

- Nalezení minimálního momentu setrvačnosti vůči jednotlivým osám hlavního souřadného systému tělesa pro každé z dvojice vázaných těles (Rovnice 39).

$$\begin{aligned} I_{1_{\min}} &= \min\left\{I_{1_x}, I_{1_y}, I_{1_z}\right\} \\ I_{2_{\min}} &= \min\left\{I_{2_x}, I_{2_y}, I_{2_z}\right\} \end{aligned} \tag{39}$$

- Určení redukované hmotnosti (Rovnice 40) a momentu setrvačnosti (Rovnice 41) vzhledem k vazebnému bodu.

$$m_{red} = \min\left\{\frac{1}{\frac{1}{m_1}+\frac{L_1^2}{I_{1_{\min}}}}, \frac{1}{\frac{1}{m_2}+\frac{L_2^2}{I_{2_{\min}}}}\right\} \tag{40}$$

$$I_{red} = \min\left\{I_{1_{\min}}, I_{2_{\min}}\right\} \tag{41}$$

Poznámka: Redukovaná hmotnost a moment setrvačnosti by měly být takové parametry fiktivního bodového tělesa, které by u něj zajišťovaly obdobnou míru odezvy z pohledu kinematických veličin jako u vazebného bodu skutečného tělesa

a to za předpokladu, že by na toto fiktivní těleso působily v těžišti stejné silové účinky, jako na vazebný bod skutečného tělesa.

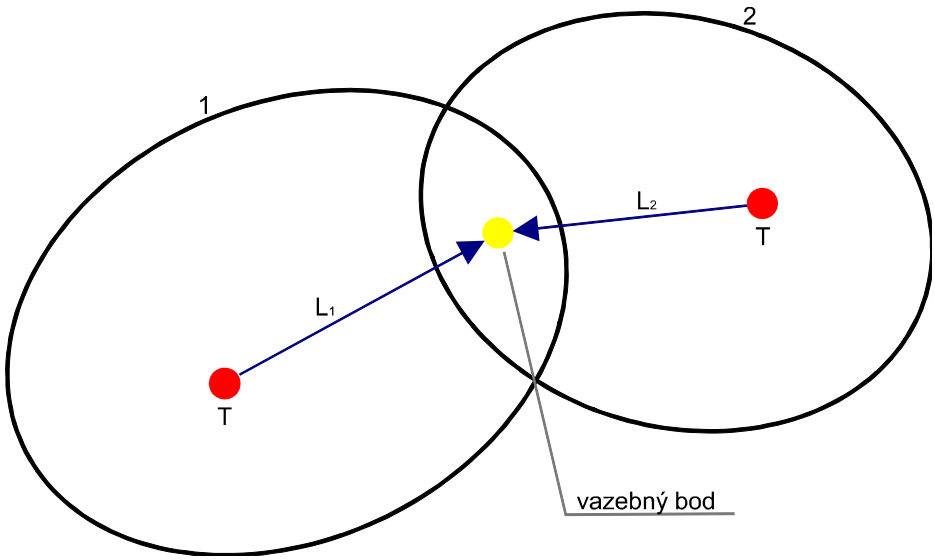
- Výpočet hlavních koeficientů pro matici translačních vazebných parametrů (Rovnice 42) a matici rotačních vazebných parametrů (Rovnice 43) podle empirických vztahů.

$$\begin{aligned} K_{t_{?x}} &= m_{red} \cdot 0.1 \\ K_{t_{?y}} &= m_{red} \cdot 5000 \\ K_{t_{?z}} &= m_{red} \cdot 800 \end{aligned} \tag{42}$$

$$\begin{aligned} K_{r_{?x}} &= I_{red} \cdot 0.09 \\ K_{r_{?y}} &= I_{red} \cdot 4500 \\ K_{r_{?z}} &= I_{red} \cdot 750 \end{aligned} \tag{43}$$

Takto získané koeficienty se poté doplní do jednotlivých řádků matic K_t a K_r podle určeného typu vazby. Řádky odpovídající definovaným stupňům volnosti vazby se vyplní nulami.

Přestože systém automatického odhadu vazebných parametrů poskytuje rychlý způsob nastavení vazeb, nemusí být vždy tyto vazby pro všechny případy mechanismů a jejich spojení ideální. Z toho důvodu je v programu zastoupena možnost definice plně uživatelských vazeb, u kterých je možno všechny vazebné koeficienty přesně určit.



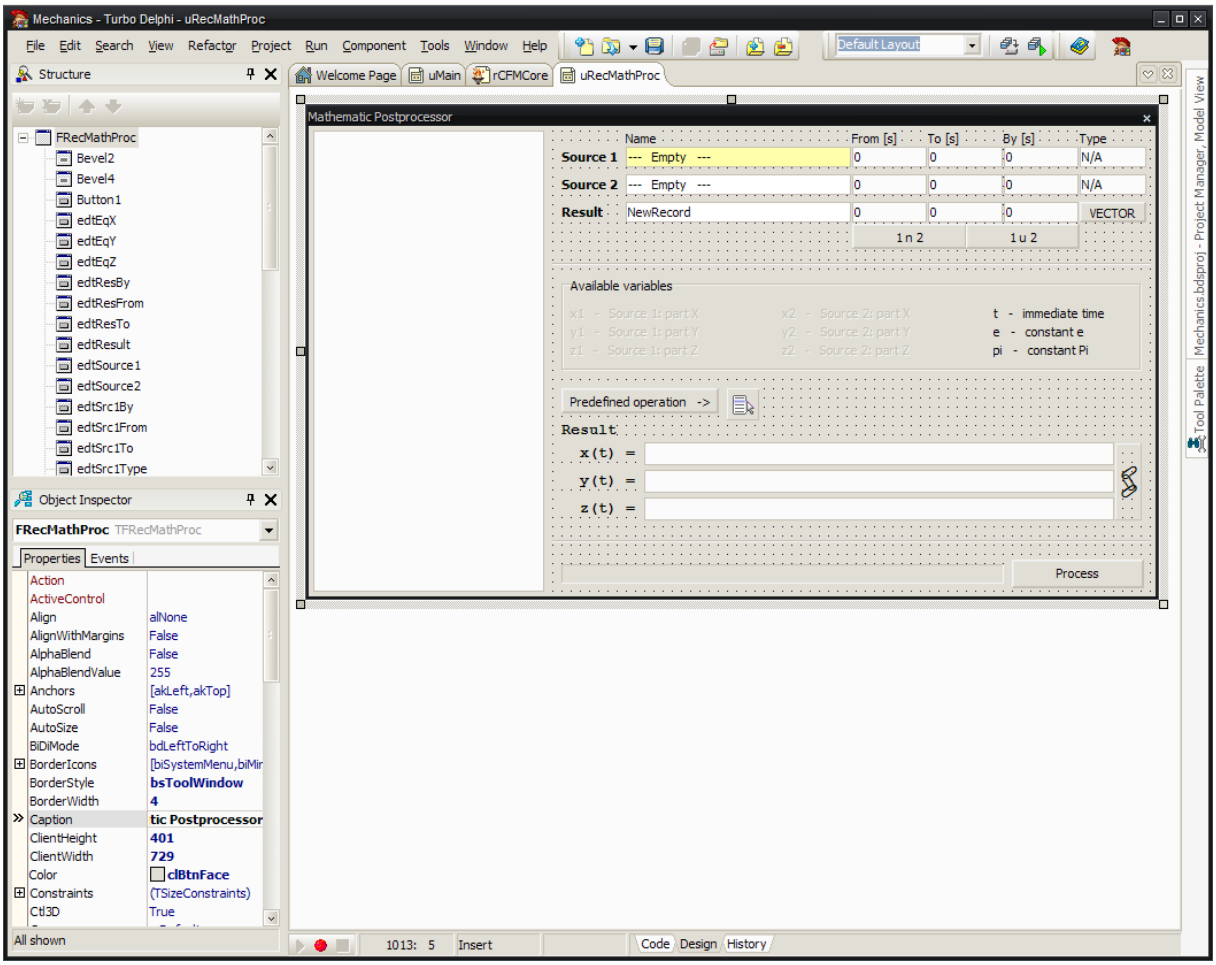
Obrázek 18: Schematické znázornění vázaných těles

6. PROGRAMOVÁ IMPLEMENTACE MKS

6.1. VÝVOJOVÉ PROSTŘEDÍ

Aplikace je kompletně naprogramována v prostředí Borland Delphi pro operační systém Microsoft Windows a platformu x86 kompatibilní. Vývoj započal roku 2005 ve verzi Delphi 6 a postupně přecházel přes novější verze až po současné Turbo Delphi.

Toto vývojové prostředí typu RAD (Rapid Application Development) bylo zvoleno z toho důvodu, že podstatně zjednodušuje rutinní programátorské úkony spojené například s tvorbou uživatelského rozhraní (Obrázek 19), a tím umožňuje programátorovi soustředit se na tvůrčí činnost a řešení složitějších problémů. Důsledkem tohoto přístupu je radikální zvyšení rychlosti vývoje aplikace a schopnost tvorby větších a složitějších projektů.

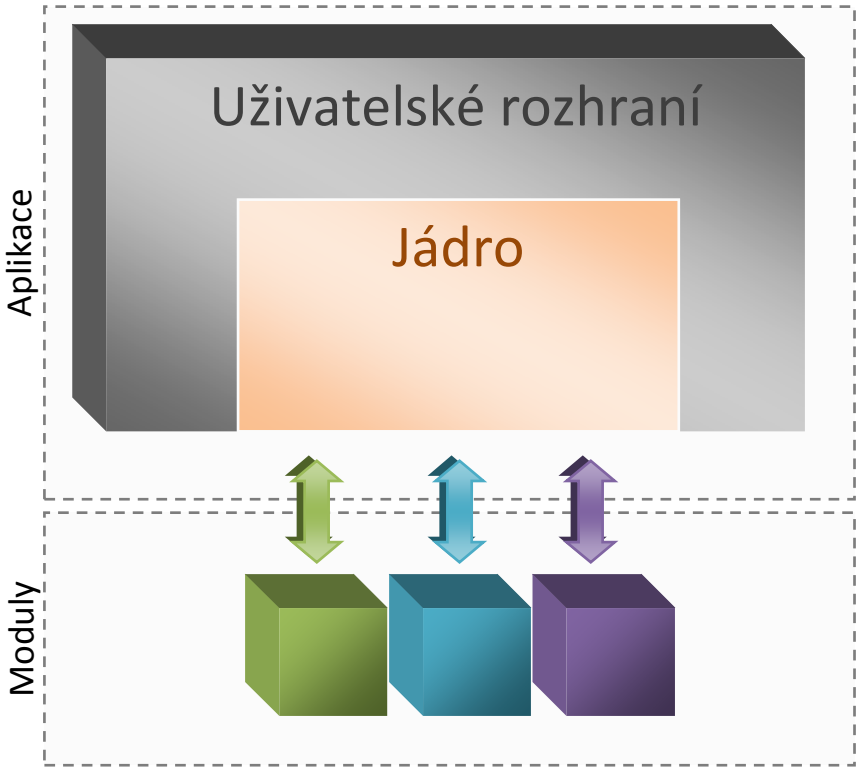


Obrázek 19: Návrh jednoho z formulářů uživatelského rozhraní aplikace Mechanics v prostředí Borland (CodeGear) Turbo Delphi

Další nespornou výhodou vývojového nástroje Borland Turbo Delphi je možnost jeho bezplatného použití i pro komerční aplikace. Jediným omezením je nemožnost instalace nových komponent přímo do prostředí Delphi. Stále však zůstává možnost užití libovolných komponent způsobem dynamického vytvoření jejich objektů za běhu programu.

6.2. STRUKTURA APLIKACE

Jak již bylo řečeno v předchozích kapitolách, principy MKS jsou implementovány v programu Mechanics. Aplikace je vytvořena moderním způsobem, kdy samotné výpočetní jádro je prakticky nezávislé na uživatelském rozhraní. Tato koncepce umožňuje případně nepříliš obtížné portování jádra na jiné operační systémy (například Linux), pro které existuje adekvátní překladač jazyka Object Pascal.



Obrázek 20: Schématické znázornění struktury aplikace

Jádro v sobě implementuje jednak samotné algoritmy metody korekčních sil, ale i další části v podobě správy parametrického stromu, systému podpory rozšiřujících výpočetních modulů, systému pro práci se záznamy simulace a dále grafického, konzolového a skriptovacího subsystému. Kolem tohoto jádra je pak vybudováno aplikační rozhraní, které umožňuje uživateli snadnou práci se všemi prostředky jádra a zpřístupňuje také mnohé další pomocné funkce, zejména v oblasti tvorby modelů, interaktivní simulace a postprocessingu (viz. Příloha č. 1).

6.3. PARAMETRICKÝ STROM

Veškerá data reprezentující aktuální model a jeho stav jsou uloženy ve stromové struktuře. Kromě samotných elementů, z nichž je složený mechanický model, jsou ve stromu i větve s vlastnostmi jádra, viewportu, rozšiřujících modulů, záznamů a widgetů.

6.3.1. DATOVÉ TYPY

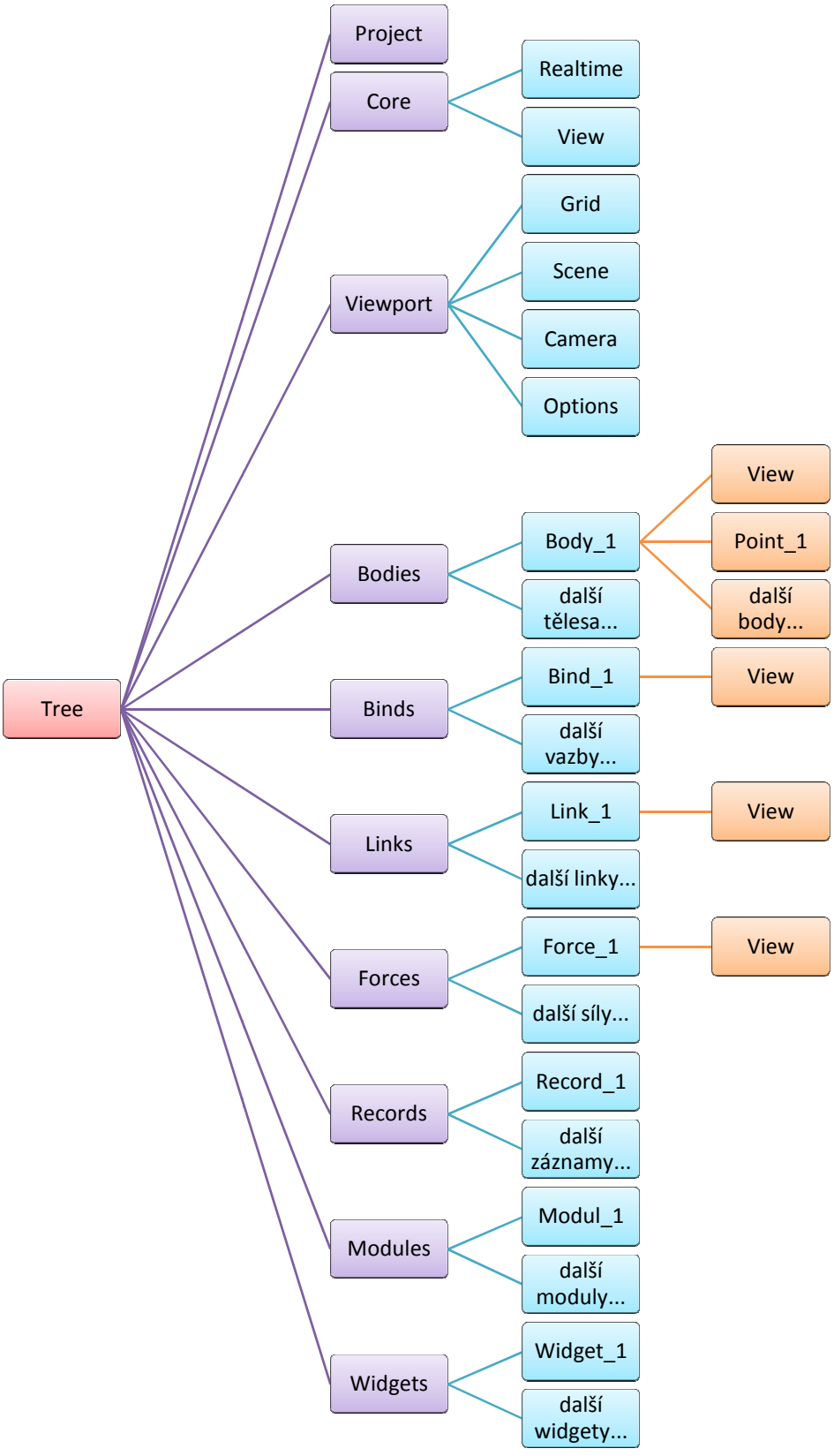
V následující tabulce (Tabulka 2) je uveden přehled typů proměnných, které jsou užívané v parametrickém stromu. Stejné typy mohou mít také parametry příkazů konzole a lze je využívat i ve skriptech konzole a rozšiřujících výpočetních modulech. Pokud je některý typ tvořen složenou strukturou, jsou jeho složky taktéž součástí parametrického stromu v další úrovni.

Typ hodnoty	**Název (zkratka)**	**Popis**	**Struktura**
Skalár	SCALAR (SCL)	Skalární hodnota z oboru reálných čísel. Maximální rozsah je -5.0 x 10^324 až 1.7 x 10^308. Počet platných míst je 15 až 16.	
Vektor	VECTOR (VEC)	Vektor se třemi složkami skalárního typu.	$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$
Matice	MATRIX (MTX)	Čtvercová matice s devíti složkami skalárního typu.	$\begin{bmatrix} xx & xy & xz \\ yx & yy & yz \\ zx & zy & zz \end{bmatrix}$
Celé číslo	INTEGER (INT)	Číselná hodnota z oboru celých čísel. Maximální rozsah je -2147483648 až +2147483647.	
Boolean	BOOLEAN (BLN)	Pravdivostní hodnota, nabývající buďto hodnoty true (ano), nebo false (ne).	
Barva	COLOR (CLR)	Vektor se čtyřmi složkami skalárního typu, užívaný pro definici barvy v klasickém aditivním barevném modelu, kde čtvrtý kanál určuje průhlednost. Všechny složky mohou nabývat reálných hodnot od 0 do 1.	$\begin{bmatrix} R \\ G \\ B \\ A \end{bmatrix}$
Řetězec	STRING (STR)	Textový řetězec. Může obsahovat libovolné znaky, kromě mezer (místo nich lze používat například podtržítko _). Maximální délka řetězce je omezena na 64 znaků.	
Reference	XXX-REF (REF)	Textový zápis reference na proměnnou příslušného typu v parametrickém stromu.	

Tabulka 2: Tabulka přehledu základních datových typů

6.3.2. STRUKTURA STROMU

Na následujícím schématu je znázorněna základní struktura stromu, kde každý samostatný prvek pod sebou sdružuje vlastní parametry a případně další prvky.



6.3.3. ZÁSOBNÍK STAVŮ

Koncentrace veškerých údajů o aktuálním modelu a jeho stavu do parametrického stromu umožňuje rychlé ukládání a obnovu těchto údajů do/z paměti. Pro využití těchto možností poskytuje jádro tzv. zásobník stavů, do kterého je možné dočasně uložit jakýkoliv aktuální stav modelu a později ho vyzvednout, jako kdyby se pracovalo se soubory uloženými na disku. Zásobník stavů je možné ovládat pomocí konzolových příkazů. Uložení i vyzvednutí libovolného stavu je prakticky okamžité.

6.4. OPTIMALIZACE JÁDRA

Jádro využívá objektový programátorský model pro správu modelu a silně optimalizované výpočetní rutiny částečně psané v assembleru (nízkoúrovňový programovací jazyk), díky čemuž dosahuje velmi vysokého výkonu (v závislosti na složitosti modelu a výkonu procesoru desítky až stovky tisíc iterací za sekundu) a minimálních nároků na kapacitu operační paměti.

Mezi další prvky a technologie, které jsou využívány pro zvýšení rychlosti patří mimo jiné:

- výkonný správce paměti (FastMM4) [experimentálně]
- zarovnávání paměťových bloků na 16 bytů [experimentálně]
- minimalizace větvení kódu

6.4.1. SIMD

Podporováno je také moderní SIMD (Single Instruction Multiple Data) rozšíření instrukční sady procesorů, které slouží k urychlení vektorových výpočtů s čísly v plovoucí desetinné čárce. Konkrétně se jedná o sadu instrukcí SSE2, nalézající se v současné době na těchto procesorech:

- Intel Pentium M
- Intel Pentium 4
- Intel Xeon
- Intel Core
- Intel Core 2 a novější…
- AMD Sempron 64
- AMD Athlon 64
- AMD Athlon FX
- AMD Opteron
- AMD Phenom a novější…

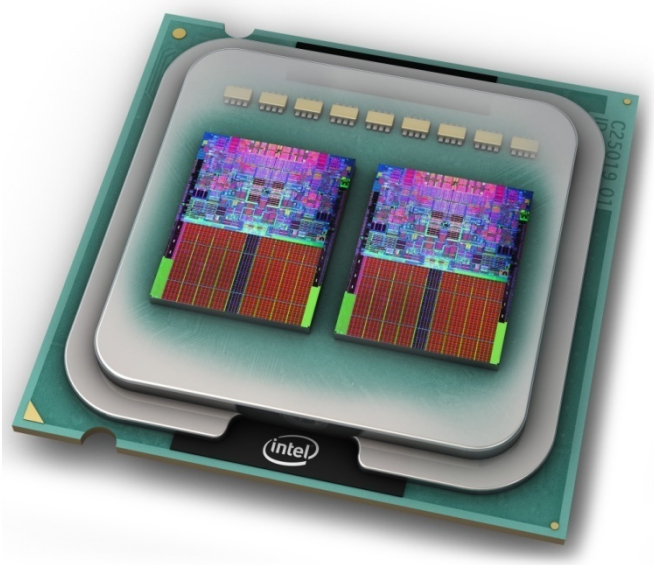
Na ostatních procesorech je dostupný pouze režim využívající klasickou floating-point instrukční sadu x87.

6.4.2. PARALELIZACE VÝPOČTŮ

Předpokladem pro rychlé vykonávání výpočetně náročného algoritmu na současných počítačích je zejména rychlý procesor. Zjednodušeně řečeno závisí výpočetní výkon každého procesoru na jeho taktovací frekvenci a počtu instrukcí, které je schopen za jeden takt vykonat.

První způsob, a to sice zvyšování výkonu procesoru prostřednictvím zvyšování jeho taktovací frekvence, naráží ovšem v dnešní době již na silné fyzikální a výrobní limity. Ve stručnosti jde o to, že se stoupající frekvencí se příliš zvyšuje spotřeba procesoru a tudíž i jeho vyzářené teplo, které je nutné odvádět.

Výrobci procesorů se tudíž vydávají čím dál více druhou cestou zvyšování výkonu, kterou je zmíněné zvyšování počtu instrukcí zpracovaných za jeden tak, čili cestou většího paralelismu. To je uskutečňováno jednak sofistikovanou architekturou samotného výpočetního jádra procesoru, ale také umísťováním více výpočetních jader do jednoho procesoru (Obrázek 21), či využitím více procesorů v jednom počítači.



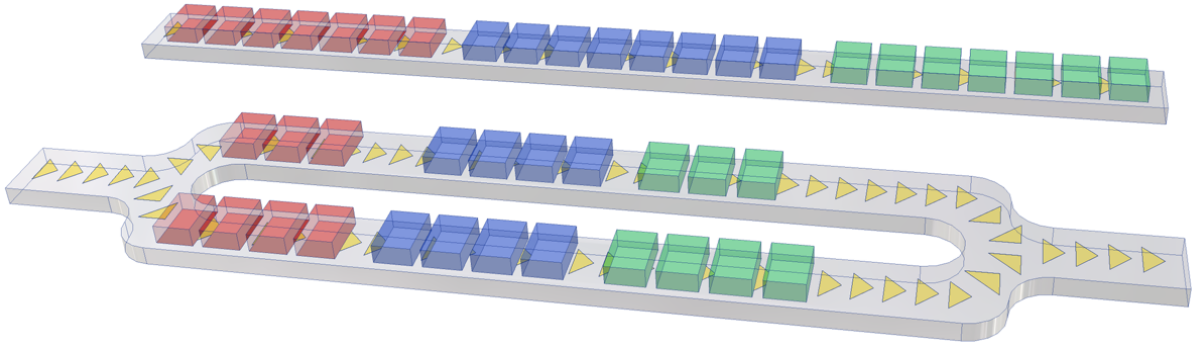
Obrázek 21: Moderní dvoujádrový procesor [7]

Proto další podporovanou moderní technologií je paralelizace výpočtů pro víceprocesorové či vícejádrové systémy. Metoda korekčních sil je z principu velice dobře paralelizovatelná, neboť elementy jednoho typu je možné počítat nezávisle na sobě v libovolném pořadí, tedy ve většině případů i paralelně.

Pokud je aktivována paralelizace výpočetního jádra, vytvoří se pro každý procesor (nebo jeho samostatné jádro) separátní programové vlákno, které zpracovává část simulovaného mechanismu. Počet takto vytvořených vláken a tedy i počet procesorů nebo jader využitelných pro paralelizaci není omezen. Jediným limitem je pouze míra efektivity. Paralelizace výpočtů se vyplatí zhruba od poměru 10 těles na jedno

výpočetní vlákno. Samozřejmě je také kontraproduktivní vytvoření více výpočetních vláken, než je dostupných procesorů či jader.

Na schématu (Obrázek 22) je znázorněn průběh výpočtů mechanismu v režimu pro jeden procesor (jedno výpočetní vlákno) a dva procesory (dvě výpočetní vlákna). Jednotlivé typy elementů jsou barevně odlišeny. Šipky naznačují směr zpracovávaných dat. U dvouvláknového režimu je znázorněno, že elementy následujícího typu (modré kostičky) se nemohou začít počítat dřív, dokud neskončí zpracování elementů předchozího typu (červené kostičky) v obou výpočetních vláknech. Z toho důvodu by mělo být rozdělení elementů každého typu pro jednotlivá výpočetní vlákna co nejrovnoměrnější a zároveň by těch elementů nemělo být příliš málo, aby relativní délka prostoje při čekání na synchronizaci nebyla zbytečně veliká a nesnižovala se tím příliš efektivita využití dostupného výpočetního výkonu.



Obrázek 22: Schematické znázornění průběhu jednovláknového a dvouvláknového výpočtu

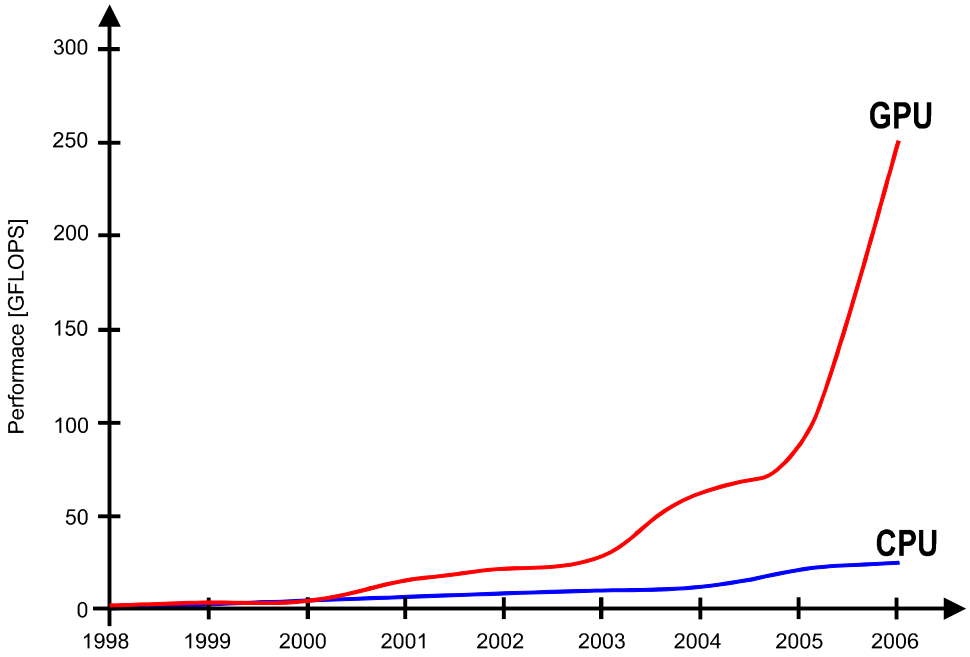
V některých případech je navíc nutné ošetřit potenciální paměťovou kolizi při výpočtu dvou různých elementů, které však přistupují k datům jednoho společného elementu (typicky dvě vazby ve stejném bodě tělesa).

Všechny tyto okolnosti (a ještě některé další) spolu s režií samotného rozdělování úloh pro více jader či procesorů způsobují, že není možné využít maximálního teoretického výkonu výpočetního systému. Přesto však celkový výkonnostní nárůst pro paralelní systémy je výrazný. Pro dvouvláknové zpracování (dva procesory nebo jádra) se nárůst výkonu pohybuje typicky mezi 50 až 95 procenty, dle typu a složitosti mechanismu.

6.4.3. GP-GPU

V budoucnu by bylo možné také využití procesoru na grafických kartách k urychlení některých výpočetních operací. Výkon moderních procesorů grafických akcelerátorů (GPU) totiž díky speciální masivně paralelní architektuře dosahuje hodnot nesrovnatelných s klasickými procesory (CPU). Současné CPU dosahují výkonu zhruba desítek GFLOPS (miliard operací reálnými čísly za sekundu), zatímco GPU se pohybují

řádově ve stovkách GFLOPS (Obrázek 23) a nové generace se mají pohybovat již okolo hranice jednoho TFLOPS (biliardy operací s reálnými čísly za sekundu), což je hodnota běžná pro superpočítače.



Obrázek 23: Srovnání růstu výkonu CPU a GPU v období let 1998 až 2006

Tento způsob využití grafických akcelerátorů pro obecné výpočty se nazývá General Purpose Graphics Processing Unit (GP-GPU).

6.5. GRAFICKÝ SUBSYSTÉM

Grafický subsystém využívá multiplatformní knihovny OpenGL (Open Graphics Library). Tato knihovna určená pro práci s hardwarově akcelerovanou 3D grafikou vznikla u společností Silicon Graphics Inc. (SGI) v roce 1992. Rozhraní OpenGL se od té doby stalo průmyslovým standardem ve zpracování prostorové grafiky a je podporováno všemi výrobci 3D grafických akcelerátorů. Na tomto rozhraní jsou postavené prakticky všechny soudobé 3D CAD/CAM/CAE systémy, animační aplikace, systémy pro virtuální realitu i některé hry.

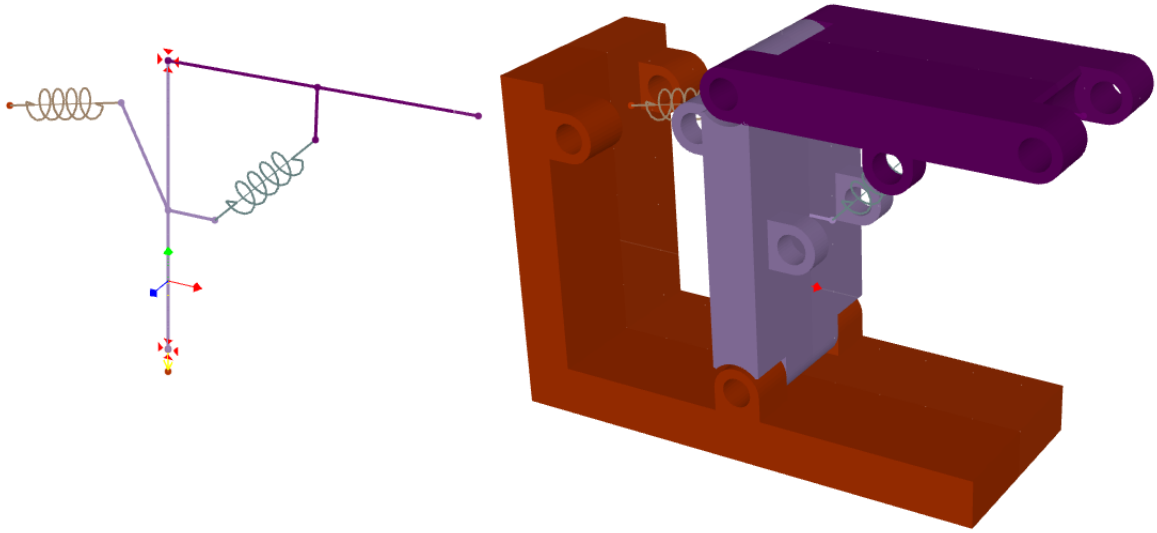
Obrázek 24: Logo rozhraní OpenGL [9]

Systém Mechanics nevyužívá OpenGL jen pro standardní zobrazování a editaci trojrozměrného modelu, ale i pro vykreslování dvourozměrných interaktivních kontrolních a ovládacích prvků. Je to z toho důvodu, že hardwarově akcelerované vykreslování pomocí OpenGL, na rozdíl od standardního Windows GDI rozhraní, nabízí vysoký grafický výkon, aniž by docházelo k nadměrnému zatěžování hlavního procesoru počítače, který se tak může maximálně věnovat simulaci jako takové.

6.5.1. IMPORT GEOMETRIE Z CAD SYSTÉMŮ

Mechanismy, resp. tělesa vytvořené v MKS systému mechanics jsou tvořeny pouze svou kostrou. To znamená, že grafické znázornění takového tělesa spočívá v tom, že jednotlivé body tělesa jsou spojeny čarami s bodem jeho těžiště (Obrázek 25).

Pro větší názornost simulovaného mechanismu umožňuje systém importování trojrozměrné geometrie ve formátu STL (StereoLitho) z CAD systémů a následné připojení této geometrie k jednotlivým tělesům mechanismu.



Obrázek 25: Zobrazení mechanismu v podobě kostry a s importovanou geometrií

Importovaná geometrie těles nijak neovlivňuje samotný průběh simulace mechanismu, jedná se skutečně pouze o vizuální prvek zvyšující názornost. Barva importované geometrie závisí na barvě, která je přiřazena tělesu, k němuž je geometrie připojena. Stejnou barvou je vykreslována i kostra tělesa, jeho body a silové výslednice v nich.

Protože importovaná geometrie nemá vliv na funkci mechanismu a není tedy ani vyžadována, ukládá se do separátního souboru – takzvané knihovny tvarů, či anglicky Shape library (od toho je odvozena koncova souborů *.shl). Výhoda uvedeného přístupu je v tom, že taková knihovna může být sdílena více mechanismy, což zrychluje práci a snižuje kapacitní nároky. Další výhodou je, že pouhým načtením jiné verze knihovny se může snadno docílit zcela jiného vnějšího vzhledu těles.

6.6. KONZOLE A SKRIPTOVÁNÍ

Nedílnou součástí jádra je i systém konzole a jejího skriptování, který poskytuje prostřednictvím několika jednoduchých příkazů široké možnosti v ovládání aplikace.

6.6.1. KONZOLE

Většinu funkcí jádra je možno ovládat pomocí textových příkazů, zadávaných do konzolového řádku (Obrázek 26), který je umístěna v okně s modelem. Nad konzolovým řádkem je integrována kontextová plovoucí nápověda, která pomáhá uživateli s orientací v typech a počtech zadávaných parametrů příkazů.

Obrázek 26: Konzolový řádek s integrovanou nápovědou

Příkazy se dají rozdělit do následujících skupin:

- příkazy pro práci s parametrickým stromem (přidávání a odebírání elementů, nastavování hodnot, hromadné úpravy, atd.)
- příkazy pro ovládání simulace (spuštění, zastavení, restart, práce se zásobníkem stavů, atd.)
- informativní příkazy (různé výpisy, statistiky, a další informace)
- pomocné příkazy (např. pro náhodné obarvení těles)
- skripty

Některé příkazy podporují v zadávaných parametrech regulární výrazy pro nahrazení jednoho nebo skupiny znaků zástupným symbolem. Díky tomu je možné jedním příkazem provádět hromadné změny v parametrickém stromu. Zástupné symboly jsou:

- * – hvězdička – nahrazuje libovolně dlouhou skupinu znaků
- ? – otazník – nahrazuje právě jeden znak

Například příkaz **„`inc bodies.kl*.center.pt.y 0.1`“** bude aplikován na všechna tělesa, jejichž název začíná slabikou „kl“, jako třeba „klika“, „kladka“, atd.

Oproti tomu příkaz **„`mul bodies.prut?8.view.size 2`“** bude aplikován na tělesa s názvem „prut18“, „prut28“, „prut38“, „prut48“, atd., neboť se nahrazuje pouze jeden znak na pozici otázníku.

Kompletní výčet všech dostupných příkazů konzole je uveden v příloze č. 2.

6.6.2. SKRIPTY

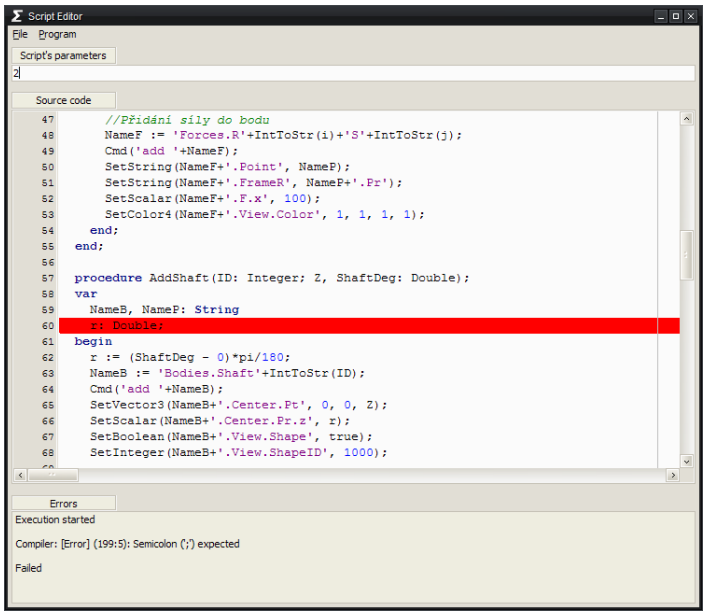
Skripty slouží k zautomatizování ovládání konzole pomocí programování. Je možné je využít například pro rychlé generování modelů v závislosti na vstupních parametrech, nebo si může uživatel napsat různé pomocné funkce pro práci s modelem (jednoduchý ukázkový skript je uveden v příloze č. 3).

Během simulace nejsou skripty vykonávány a není tudíž možné pomoci těchto skriptů řídit samotný průběh simulace. K tomu slouží rozšiřující výpočetní moduly.

Jazykem využitým pro skriptování je Pascal. Je tedy možné využívat všechny jeho standardní konstrukce, jako jsou například:

- begin/end
- if/then/else
- for/to/downto/do
- repeat/until
- while/do
- case x of
- exit/continue/break
- standardní datové typy (včetně typů z parametrického stromu)
- funkce a procedury

Skripty je možné psát v libovolném textovém editoru, nebo využít s výhodou integrovaného nástroje pro psaní skriptů, který nabízí zvýrazňování syntaxe a zobrazení chyb ve zdrojovém kódu (Obrázek 27).



Obrázek 27: Editor skriptů s vyznačeným řádkem, na kterém došlo k chybě

6.7. ROZŠIŘUJÍCÍ MODULY

Jádro systému poskytuje jednoduché programové rozhraní, umožňující snadné rozšíření výpočetních schopností pomocí zásuvných modulů reprezentovaných formou DLL knihoven. Ty mohou být psány prakticky v libovolném programovacím jazyce, který technologii DLL knihoven podporuje.

Rozhraní je tvořeno následujícími prvky:

- funkce pro (de)registraci vybraných proměnných modulu do/z parametrického stromu
- funkce pro (de)registraci exekutivních procedur pro zpracování událostí vyvolaných jádrem
- pomocné funkce
- definice datových typů a struktur užívaných jádrem

Takto vytvořený modul může být plnohodnotnou součástí parametrického stromu a bez omezení přistupovat ke všem jeho položkám, stejně jako nástroje integrované přímo v aplikaci. Přístup k datům je řešen pomocí ukazatelů (pointerů), čili nedochází ke zbytečnému kopírování dat mezi jádrem a modulem.

Pomocí rozšiřujících modulů je možno řešit řadu speciálních případů v simulaci mechanismů, které samotné jádro schopné řešit není, neboť je koncipováno způsobem, který staví na co největší obecnosti, zaručujícím vysokou variabilitu a zároveň jednoduchost systému. Typickým polem pro využití výpočetních modulů je řešení různých matematických závislostí mezi vstupními a výstupními veličinami modelu. Jeden z dodávaných výpočetních modulů například simuluje průběh tlakové síly na píst ve spalovacím motoru.

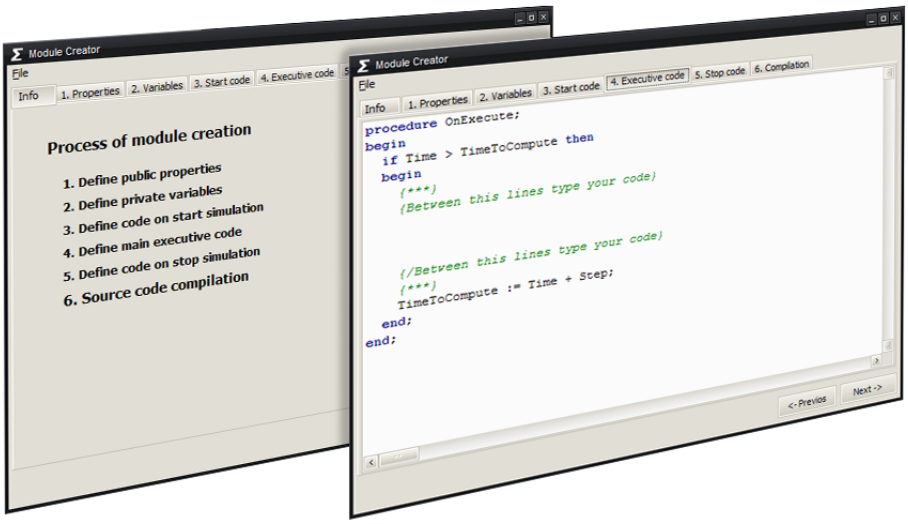
Modulární systém zároveň nabízí vysokou výpočetní rychlost, neboť veškeré moduly jsou předem zkompilované a tudíž mnohem rychlejší, než kdyby byly řešeny například formou interpretovaných skriptů.

Jedinou nevýhodou rozšiřujících výpočetních modulů je fakt, že není možné je na víceprocesorových systémech automaticky vykonávat paralelně na úrovni jádra, neboť nelze zajistit ochranu proti paměťovým kolizím. Případná paralelizace výpočtů uvnitř nějakého složitého modulu samozřejmě možná je, ale musí se o ni postarat sám tvůrce modulu.

Počet instancí výpočetních modulů zavedených do jádra z jedné DLL knihovny není omezen.

6.7.1. INTEGROVANÝ PRŮVODCE PROGRAMOVÁNÍM MODULŮ

Pro tvorbu jednodušších rozšiřujících modulů není potřeba vlastnit žádný vývojový programovací nástroj, nebo se učit psát konstrukce nutné ke zvládnutí programování DLL knihoven pro systém Mechanics. Aplikace totiž nabízí integrovaný programovací nástroj (Obrázek 28), který formou průvodce pomůže uživateli s tvorbou jednoduchých modulů.



Obrázek 28: Integrovaný nástroj pro snadnou tvorbu modulů

Výhodou tohoto nástroje je zejména to, že oprošťuje uživatele od nutnosti psát kompletní zdrojový kód DLL knihovny, ale nechá ho jednoduchým způsobem doplnit pouze několik jeho podstatných částí. Kompletní výsledný zdrojový kód se poté automaticky vygeneruje a zkompiluje do bezprostředně použitelné DLL knihovny (pokud se kompilace nezdaří, je uživatel informován chybovým výpisem o možných příčinách).

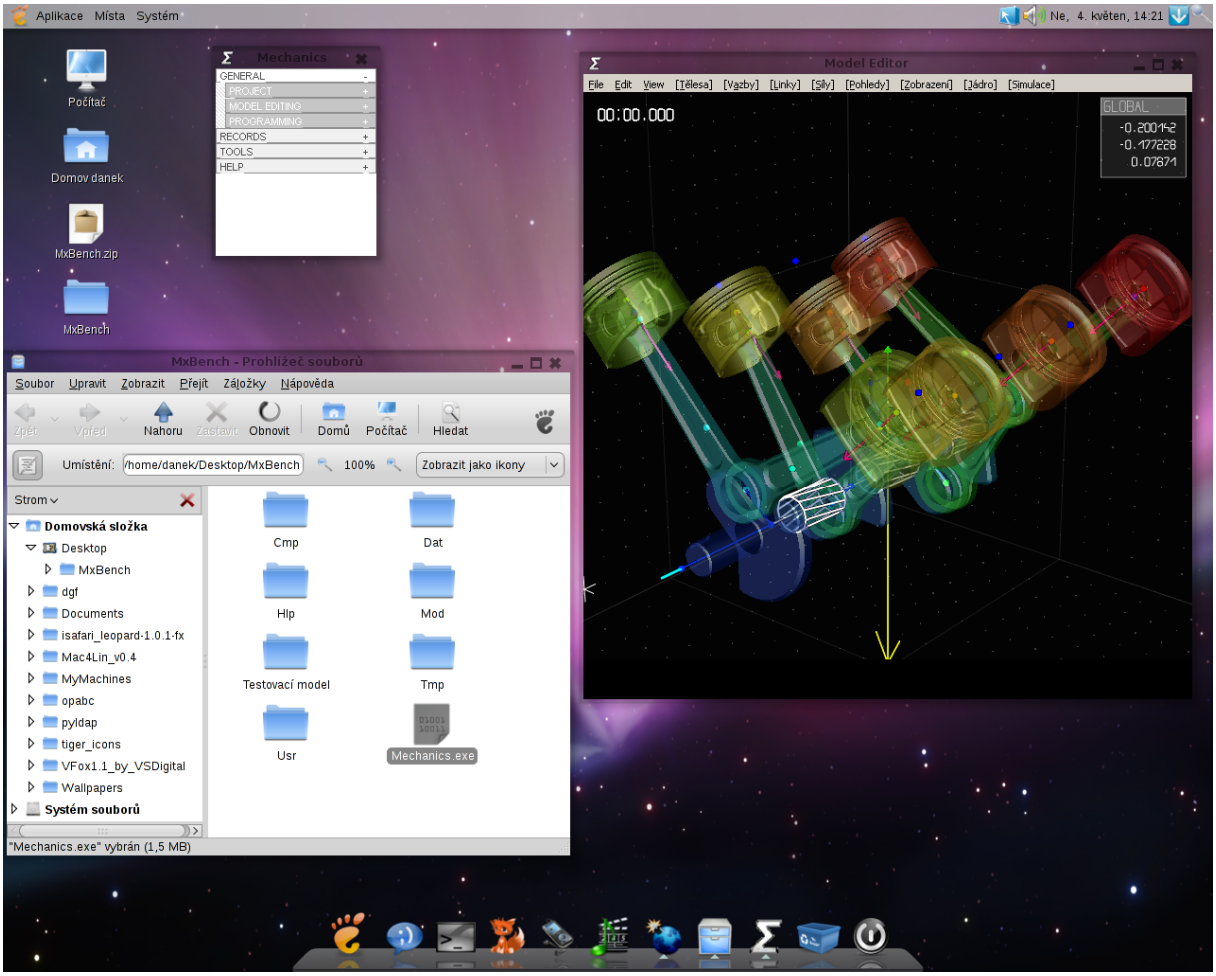
Použitým programovacím jazykem je stejně jako u skriptovacího nástroje pro ovládání konzole Pascal (kompilovaný volně dostupným open source kompilátorem Free Pascal Compiler). Aplikace Mechanics tento jazyk využívá z toho důvodu, že je velice názorný a jednoduchý na pochopení (proto se také často vyučuje na školách), ale přitom dostatečně mocný pro zvládnutí prakticky všech obvyklých úloh.

6.8. MOŽNOSTI BĚHU POD LINUXEM

Přestože je aplikace Mechanics psána výhradně pro platformu Win32, dokáže běžet i na UNIXových operačních systémech díky projektu WINE (Obrázek 29). Autoři tohoto projektu definují WINE jako překladovou vrstvu a zavaděč umožňující běh Windows aplikací na Linuxu nebo jiném operačním systému kompatibilním s normou POSIX.

Díky tomu, že WINE neemuluje celý virtuální stroj (viz. rekurzivní název programu WINE – Wine Is Not Emulator), ale zjednodušeně řečeno překládá pouze volání systémových funkcí WinAPI na linuxové systémové funkce, není pokles výkonu simulačního jádra nijak výrazný a pohybuje se zhruba v rozmezí pěti až deseti procent.

Ze stejného důvodu je ovšem běh Windows aplikací pod Linuxem prostřednictvím WINE omezen pouze na hardwarovou platformu x86 kompatibilní.

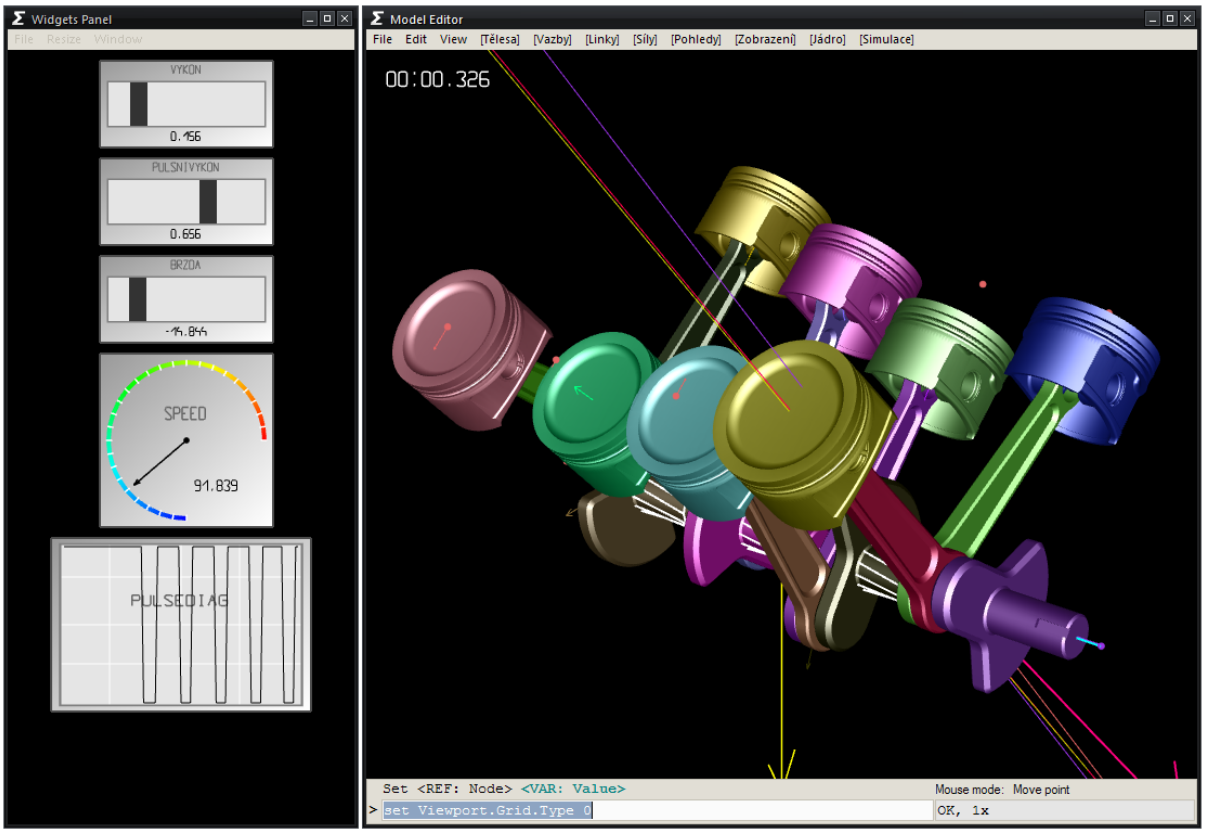


Obrázek 29: Aplikace Mechanics spuštěná v operačním systému ArchLinux

7. INTERAKTIVITA

MKS systém Mechanics umožňuje během simulace přímo měnit libovolné číselné parametry modelu (včetně parametrů rozšiřujících výpočetních modulů) a tím bezprostředně ovlivňovat jeho chování. Z toho plyne, že simulace je plně interaktivní a je tudíž možné, aby do ní uživatel v reálném čase zasahoval.

Ke každému modelu je možné přiřadit skupinu ovládacích a indikačních prvků - tzv. widgetů (Obrázek 30), které se zobrazují v samostatném okně (což je výhodné, pokud uživatel disponuje dvěma monitory připojenými k počítači, neboť pak může mít například na jednom monitoru zobrazenu v maximální velikosti scénu s modelem a na druhém všechny ovládací prvky, včetně aplikačního rozhraní).



Obrázek 30: Panel s ovládacími prvky ovlivňujícími chování simulovaného modelu

Jak již bylo zmíněno v kapitole věnující se grafickému subsystému, jsou i tyto části uživatelského rozhraní vykreslovány pomocí OpenGL, což zaručuje vysokou rychlost a nízké systémové nároky. Navíc grafická implementace widgetů je vektorové povahy, tudíž je možné libovolně měnit jejich velikost a umístění bez vlivu na kvalitu vykreslování.

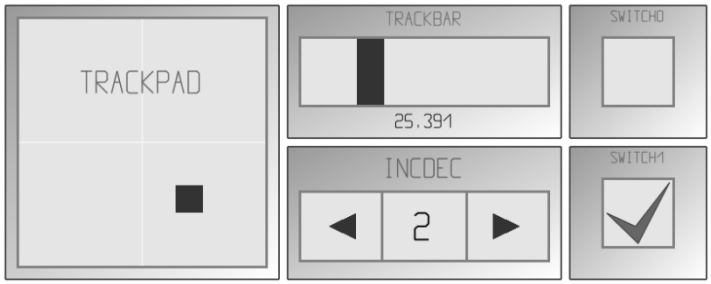
Widgety a jejich parametry jsou zároveň součástí parametrického stromu a lze tedy k jejich tvorbě či nastavení použít mimo jiné i příkazy konzole.

Uživatelem vytvořená skupina widgetů se vždy ukládá do jednoho souboru spolu s modelem. Při načítání souboru, který tyto prvky obsahuje, se automaticky zobrazí příslušné okno, neboli ovládací panel (Widgets Panel).

7.1. OVLÁDACÍ PRVKY

První skupinu widgetů dostupných v současné verzi aplikace Mechanics tvoří ovládací prvky (Obrázek 31), které slouží k přímému nastavování veličin různého typu v průběhu simulace. Patří mezi ně:

- Track Pad – ovládá dvě složky vektoru
- Track Bar – ovládá skalární hodnotu
- IncDec – ovládá celočíselnou hodnotu
- Switch – ovládá hodnotu typu boolean



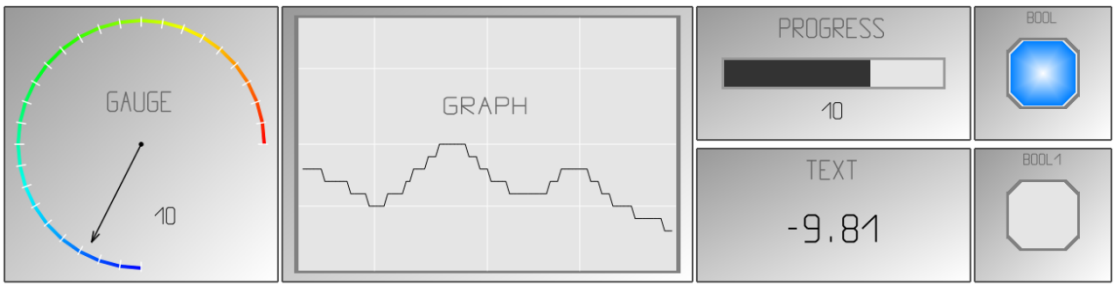
Obrázek 31: Přehled ovládacích prvků

Uživatel může všechny tyto prvky ovládat kdykoliv během běžící i neběžící simulace v jejich aktivní oblasti prostřednictvím myši připojené k počítači.

7.2. INDIKAČNÍ PRVKY

Druhou skupinu widgetů tvoří indikační prvky (Obrázek 32), jež slouží k rozličnému způsobu zobrazování veličin různého typu v průběhu simulace (ovšem reagují obdobně jako ovládací widgety na změny referenčních hodnot i mimo simulační režim). Patří mezi ně:

- Analog Gauge – zobrazuje skalární hodnotu na ručičkovém ukazateli
- Simple Graph – zobrazuje časový průběh skalární hodnoty v grafu
- Progress Bar – zobrazuje skalární hodnotu v průběhovém ukazateli
- Text Panel – zobrazuje skalárni veličinu v textovém panelu
- Indicator – zobrazuje stav hodnoty typu boolean



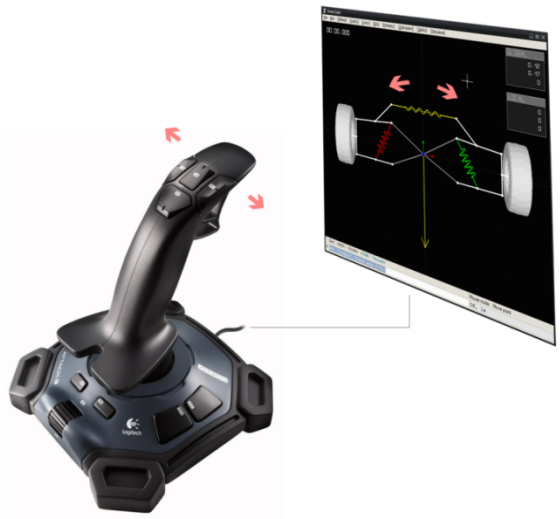
Obrázek 32: Přehled indikačních widgetů

Překročí-li u indikačních i ovládacích prvků referenční hodnota definované meze (popř. není-li reference na tuto hodnotu validní), dojde k indikaci tohoto aktu zčervenáním aktivní části widgetu.

Poznámka: Přestože většina indikačních prvků zobrazuje skalární hodnotu, je možné jako vstupy použít i vektorové proměnné. V tom případě bude zobrazována skalární hodnota velikosti vektoru.

7.3. DALŠÍ MOŽNOSTI

Multi-body systém Mechanics poskytuje i další způsoby interaktivního ovládání simulace. Jednou z nich je využití připojených externích zařízení, jako například joysticku (Obrázek 33).



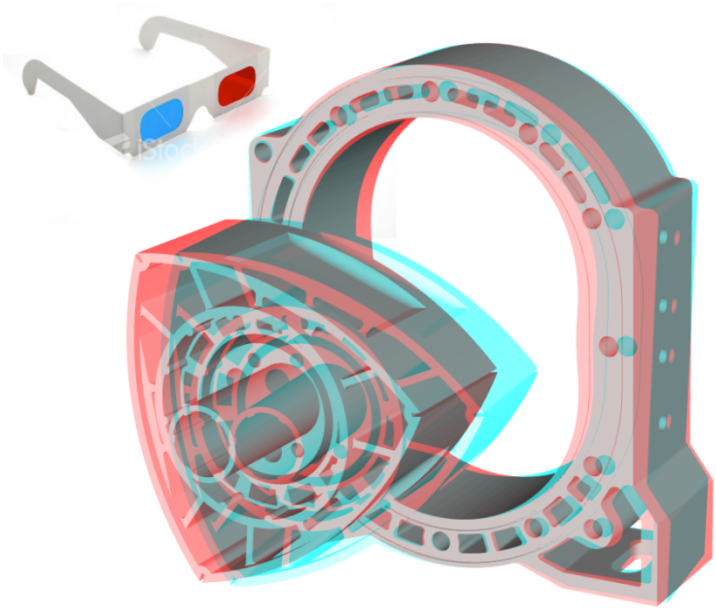
Obrázek 33: Ovládání simulovaného modelu pomocí joysticku

V tomto případě lze dle typu připojeného joysticku pomocí jeho hlavních os řídit spojitě dvě až tři referenční hodnoty a s využitím zbývajících tlačítek až další čtyři referenční hodnoty (ovšem pouze dvoustavově).

8. VIRTUÁLNÍ REALITA

Dalším prvkem spojeným s pokročilým grafickým systémem a vysokou interaktivitou aplikace Mechanics je možnost zobrazení scény se simulovaným mechanismem v režimu virtuální reality. To znamená, že uživatel sledující takto promítanou scénu má skutečný dojem prostoru a vnímání hloubky. Zobrazované objekty mohou jakoby vystupovat ze zobrazovacího média (monitor, projekční plátno, atd.), nebo se nacházet i za ním. U klasického zobrazení, které je pouze průmětem trojrozměrné geometrie na dvourozměrnou plochu, tento dojem uživatel nikdy nemá.

V současné době se pro tento typ 3D zobrazení používají nejčastěji tři postupy. Jsou jimi aktivní nebo pasivní stereoskopie a anaglyfní stereoskopie. První dva uvedené způsoby se využívají například ve známých kinech IMAX 3D.



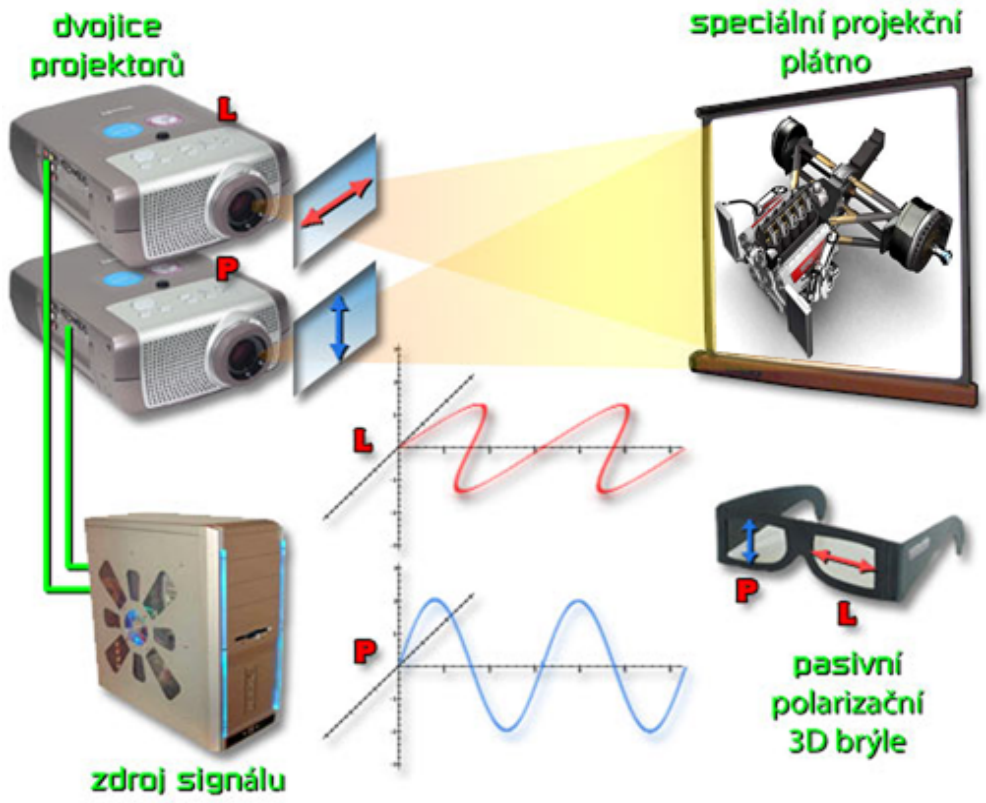
Obrázek 34: Ukázka zobrazení modelu z aplikace Mechanics v 3D stereoskopickém anaglyfním režimu

Text a ilustrace následujících kapitol (8.1 až 8.3) vycházejí částečně z webových stránek společnosti GALI-3D [6].

8.1. PASIVNÍ STEREOSKOPIE

Pasivní 3D projekce (Obrázek 35) je založena na brýlích, které mají v očnicích polarizační filtry. Jedna očnice má polarizační filtr orientovaný tak, že propouští pouze světlo kmitající v horizontální rovině. Druhá očnice obsahuje stejný o devadesát stupňů otočený filtr. Tedy takový, že propouští pouze světlo kmitající ve vertikální

rovině. Dva obrazy se promítají na jednu projekční plochu, přičemž před každým projektorem je upevněn taktéž polarizační filtr. Nastavení filtrů na projektoru koresponduje s nastavením filtrů na brýlích. Dvojice obrazů (pro pravé a levé oko) se následně promítá na jednu projekční plochu, která je vyrobena ze speciálního materiálu a opatřena povrchem, který zachová polarizaci dopadajícího světla. Odražené obrazy od projekční plochy se dostávají k divákovi, nicméně do každého oka pronikne (díky polarizačním filtrům v očnicích) pouze příslušný obraz.



Obrázek 35: Schéma pasivní stereoskopické 3D projekce [6]

8.1.1. Výhody

Kvalitní obraz ve vysokém rozlišení. Vhodné pro větší počet lidí (metodu používají standardní kina IMAX 3D). Velmi stabilní obraz. Grafický hardware nemusí být tak výkonný jako u aktivní stereoskopie.

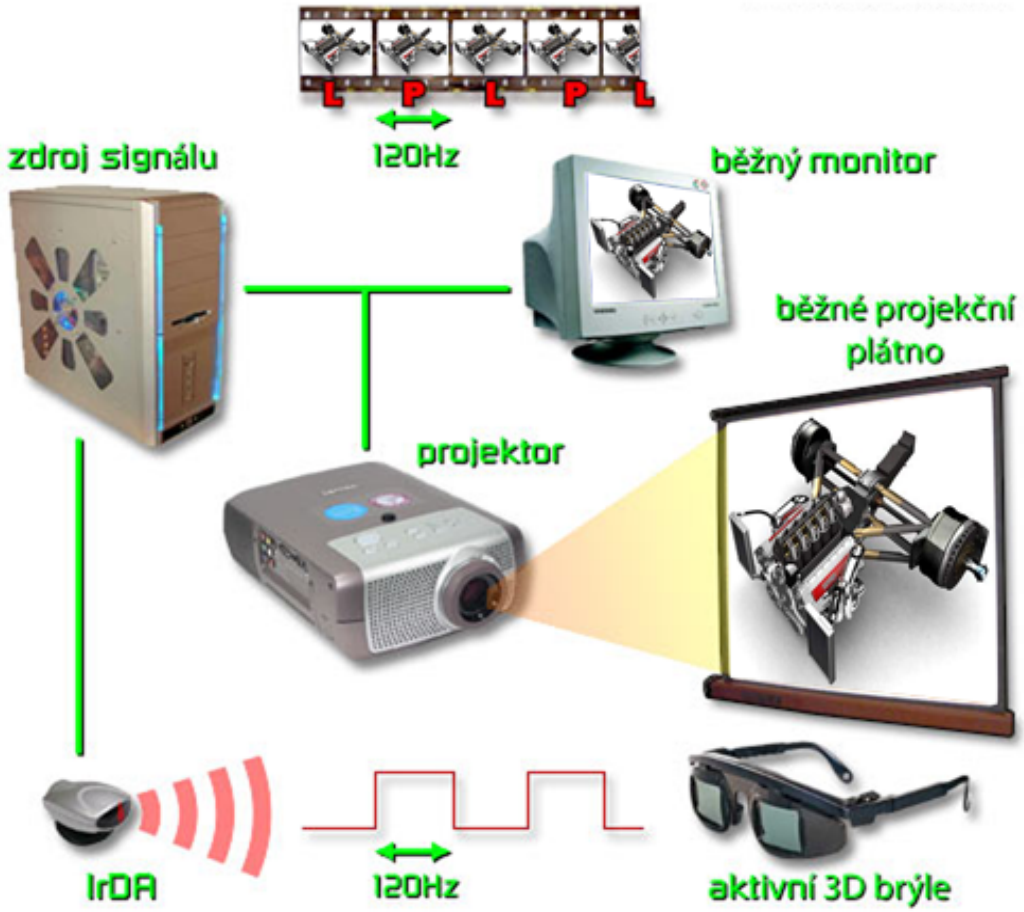
8.1.2. Nevýhody

Jsou zapotřebí dva projektory a je také nezbytně nutné speciální projekční plátno s nedepolarizujícím povrchem. Pro pasivní projekci nelze použít monitor. Relativně vysoké pořizovací náklady.

Poznámka: Projekční místnost s technologií pasivní stereoskopie je umístěna i na fakultě strojního inženýrství VUT Brno a systém Mechanics ji plně podporuje.

8.2. AKTIVNÍ STEREOSKOPIE

Vyjma pasivní stereoskopické technologie, která využívá polarizační brýle existuje i další způsob jak zajistit, aby pozorovatel obdržel do každého oka příslušný snímek. Diváci sledují obraz, který se promítá na plátno (nebo monitor) s dvojnásobnou snímkovou frekvencí (Obrázek 36), přičemž na filmovém pásu jsou střídavě proložené obrazy pro levé a pravé oko. Elektronické brýle diváka se dálkově (většinou s pomocí IrDA paprsku, nebo kabelem) synchronizují se zdrojem vysílání a střídavě zatmívají levé nebo pravé oko. Výsledkem je, že každý lichý snímek vidí uživatel jedním okem a každý sudý okem druhým. Tímto systémem se sice sníží frekvence promítaných obrazů pro každé oko na polovinu, nicméně každé oko uživatele dostává pouze svůj předurčený obraz. Z dvojice oddělených snímků mozek následně skládá skutečnou trojrozměrnou scénu. Z důvodu prokládání obrazu musí být obnovovací frekvence promítaného děje poměrně vysoká, aby nedocházelo k blikání a rychlé únavě očí. Minimální rozumná hranice je 120 Hz, lépe však 160 Hz (z toho důvodu není možné pro tento způsob zobrazení použít LCD monitory, neboť ty tak vysoké snímkové frekvence neumožňují).



Obrázek 36: Schéma aktivní stereskopické 3D projekce [6]

8.2.1. VÝHODY

Kvalitní plnobarevné zobrazení. Pro projekci stačí normální projekční plátno. Metoda také funguje velmi dobře i s kvalitními CRT monitory (ovšem LCD monitory použít nelze, neboť nedosahují dostatečné obnovovací frekvence pro nerušené sledování).

8.2.2. NEVÝHODY

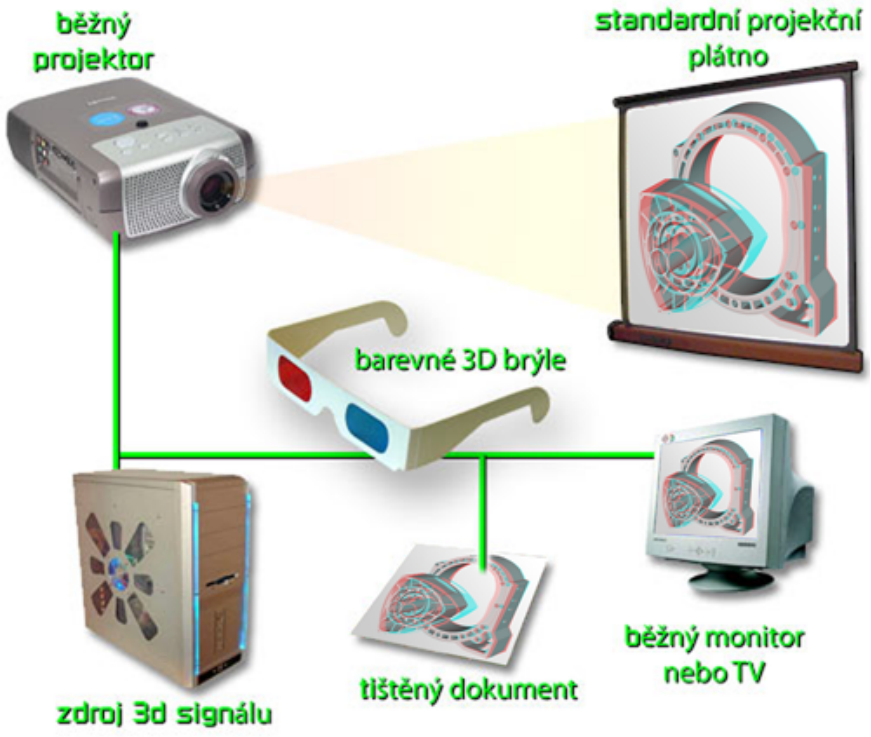
Každý uživatel musí mít poměrně komplikované elektronické brýle – není vhodné pro velký počet lidí. Projekce vyžaduje specializovaný nestandardní projektor. Pro dosažení požadované obnovovací frekvence je zapotřebí velmi výkonný grafický hardware. Pořizovací náklady jsou taktéž poměrně vysoké.

8.3. ANAGLYFNÍ STEREOSKOPIE

Anaglyfní projekce (Obrázek 37) je dnes jednou z nejvíce rozšířených metod, jak lze zobrazit 3D prostorové obrázky či případně i animace a film. Známý je především proto, že je velmi snadné zajistit jeho projekci. Ve výsledku stačí pouze brýle, které jsou vybavené jednou červenou a jednou modrou nebo zelenou očnicí. Nepsaným pravidlem je, že levé oko je vždy zabarveno červeným filtrem. Pravá očnice je potom vybavena zeleným nebo častěji modrým filtrem. Existují však i další experimentální druhy anaglyfů.

Sledovaná 3D scéna je vyrobena tak, že obsahuje smíchané oba stereoobrazy v sobě, pouze základní dvojice barev (červená a modrá nebo červená a zelená) slouží pro oddělení dvou obrazů. Pokud divák sleduje scénu s 3D brýlemi, do každého oka více či méně dostává (díky příslušným barevným filtrům) separátní obraz. Mozek ve výsledku vygeneruje z těchto obrazů 3D scénu.

Bohužel, cenou za snadné a finančně nejméně náročné zobrazení anaglyfu se platí ztrátou barevných informací. Situace je o to komplikovanější, že divák vidí scénu každým okem zcela barevně jinak (jedním okem červeně a druhým modře nebo zeleně). Mozek diváka se sice tyto ruchy snaží co možná eliminovat, ale vjem nikdy není tak kvalitní jako u jiných typů 3D projekcí.



Obrázek 37: Schéma anaglyfní stereoskopické 3D projekce [6]

8.3.1. Výhody

Technická a finanční nenáročnost, snadné přehrávání nebo prohlížení 3D scén, animací a videa na všech běžných médiích (tisk, video, projekce, PC, atd.). Při využití jako zobrazovacího režimu na PC nemusí být grafický hardware tak výkonný jako u aktivní stereoskopie.

8.3.2. Nevýhody

Scéna je barevně deformovaná a vjem není pro diváka nikdy tak poutavý jako např. u aktivní či pasivní 3D projekce. U citlivějších jedinců může docházet po delším sledování takto zobrazené scény i k bolestem očí nebo hlavy.

Metoda	Barevná informace	Rozlišení	Vhodné pro projekci	Zobrazení na monitoru	Počet diváků	Náklady
Anaglyfní zobrazení	kompletní ztráta	střední	ano	ano	vysoký	velmi nízké
Aktivní zobrazení	plná	vysoké	ano	ano (ne LCD)	omezený	vyšší
Pasivní zobrazení	plná	vysoké	ano	ne	vysoký	střední

Tabulka 3: Shrnutí vlastností metod stereoskopického zobrazení

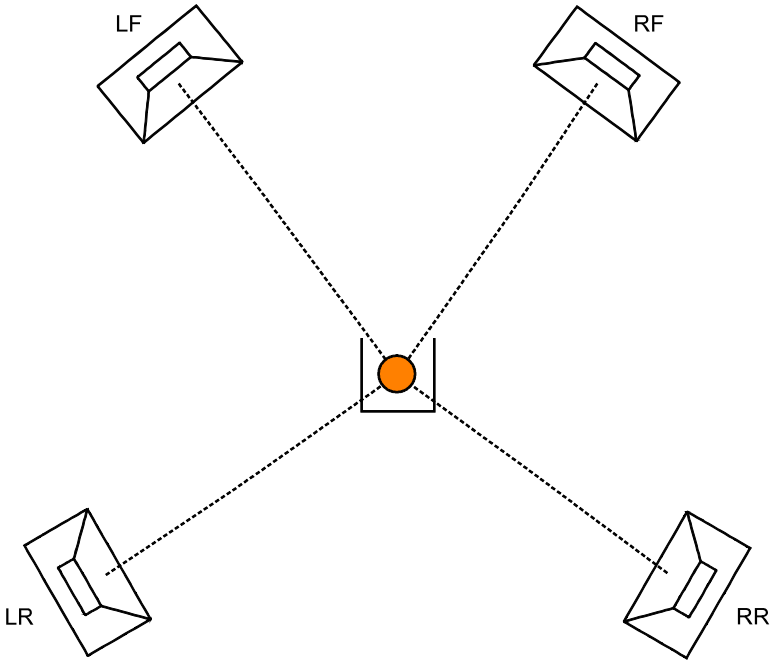
8.4. PROSTOROVÉ OZVUČENÍ

Pro zdokonalení požitku ze systému virtuální reality obsahuje aplikace Mechanics i možnost prostorového ozvučení scény. Tato technologie není ovšem přímo součástí jádra, ale je dodávána jako rozšiřující výpočetní modul.

Modul zahrnuje celkem tři parametricky řízené způsoby práce se zvukovým vzorkem, a to sice:

- frekvenční modulace
- amplitudová modulace
- umístění zvuku v prostoru

Pro zpracování zvuku využívá modul knihovny multiplatformního rozhraní OpenAL (Open Audio Library), což je taková obdoba knihoven OpenGL pro zvuk zpracovávaný pomocí zvukových karet.



Obrázek 38: Znázornění audiosystému v konfiguraci 4.0 pro prostorové ozvučení

K dosažení kýženého efektu prostorového ozvučení je zapotřebí, aby počítač obsahoval zvukovou kartu s podporou OpenAL rozhraní a připojenou reproduktorovou soustavou minimálně se dvěma kanály (konfigurace 2.0 nebo 2.1), lépe však s více (např. konfigurace 4.0, 5.0, 5.1 nebo 7.1).

9. VERIFIKAČNÍ ÚLOHY

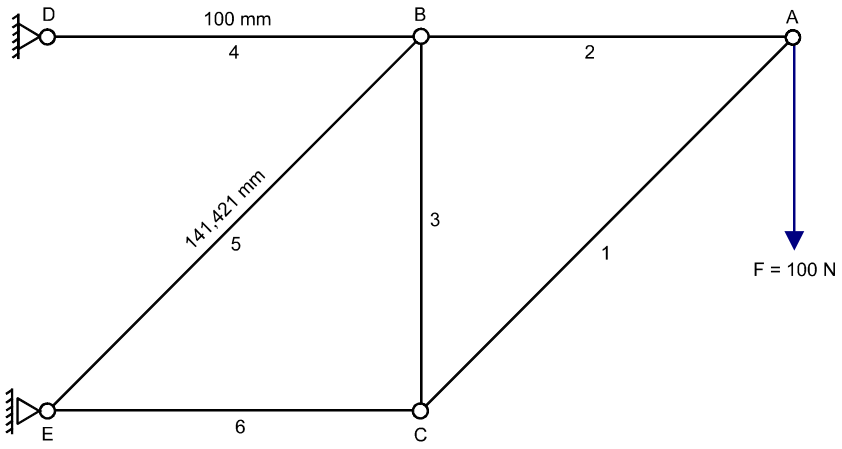
Pro ověření správnosti řešení poskytovaného systémem Mechanics bylo uskutečněno několik srovnávacích úloh oproti analytickému řešení z oblasti statiky, kinematiky a dynamiky.

9.1. ÚLOHA Č. 1 – PRUTOVÁ SOUSTAVA

První verifikační úlohou je jednoduchá rovinná prutová soustava, která se běžně řeší ve statice.

9.1.1. ZADÁNÍ

Prutovou soustavu dle náčrtu (Obrázek 39) tvoří čtyři vodorovné pruty o délce 100 mm a dva šikmé pruty o délce 141,421 mm. Úhel mezi vodorovnými a šikmými pruty je 45°. Ve spojích (styčnících) jsou rotační vazby. Na pravý konec soustavy působí vnější svislá síla o velikosti 100 N.



Obrázek 39: Prutová soustava

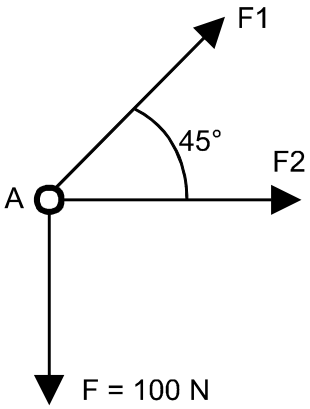
Cílem výpočtu je určení osových sil působících ve všech prutech soustavy a následné porovnání s výsledky poskytnutými numerickou simulací v systému Mechanics.

9.1.2. ANALYTICKÉ ŘEŠENÍ

K analytickému řešení uvedené prutové soustavy byla využita postupná styčníková metoda. Ta spočívá v postupném uvolňování jednotlivých styčníků a dopočítávání neznámých parametrů z rovnic statické rovnováhy.

Níže (Obrázek 40) je uvedena ukázka výpočtu pro styčník A. V tomto styčníku působí dvě osové síly (F_1 a F_2) a jedna vnější síla (F). Směr neznámých osových sil se volí směrem ven z prutu (kladná síla pak odpovídá tahovému namáhání prutu, zatímco

záporná síla odpovídá tlakovému namáhání). Všechny působící síly se dosadí do rovnic statické rovnováhy a ze vzniklé soustavy dvou rovnic se dopočítají neznámé síly F_1 a F_2.



$$\sum F_x = 0 \rightarrow F_2 + \cos(\alpha) \cdot F_1 = 0$$
$$\sum F_y = 0 \rightarrow \sin(\alpha) \cdot F_1 - F = 0$$

$$F_1 = \frac{F}{\sin(\alpha)} \doteq 141{,}421 \text{ [N]}$$

$$F_2 = -\cos(\alpha) \cdot F_1 = -100 \text{ [N]}$$

Obrázek 40: Ukázka výpočtu neznámých sil ve styčníku A

Obdobně by se dále pokračovalo v řešení sil ve zbývajících styčnících B až E. Pořadí řešení styčníků se volí podle pravidla maximálně dvou neznámých parametrů ve styčníku.

9.1.3. Numerické řešení

Systém Mechanics nepodporuje samostatné řešení čistě statických či kinematických úloh. Statickou úlohu je však možné bez potíží řešit jako dynamickou s tím, že se po spuštění simulace pouze vyčká na ustálení výsledků. V simulovaném mechanismu prutové soustavy došlo k ustálení s přesností na tři desetinná místa asi po 0,8 sekundy simulačního času. Dobu ustálení ovlivňují zejména koeficienty tlumení vazeb.

9.1.4. Srovnání výsledků

Výsledky analytického řešení i numerického řešení pomocí MKS systému Mechanics jsou shrnuty v následující tabulce (Tabulka 4). Porovnání zahrnuje velikost osové síly každého prutu a úhel sklonu její nositelky.

Odchylky sil a úhlů		Analytické řešení		Numerické řešení		Rozdíly	
		|F| [N]	α [°]	|F| [N]	α [°]	|ΔF| [N]	|Δα| [°]
Osové síly	F_1	141,421	45	141,419	44,995	0,002	0,005
	F_2	100	0	100,006	0,006	0,006	0,006
	F_3	100	90	100,002	89,996	0,002	0,004
	F_4	200	0	200,005	-0,006	0,005	0,006
	F_5	141,421	45	141,419	44,996	0,002	0,004
	F_6	100	0	99,999	-0,007	0,001	0,007

Tabulka 4: Srovnání výsledků

Z porovnání je vidět, že chyby se objevují až na pátém či šestém platném místě. Úpravou vazebných koeficientů by bylo možné tyto chyby ještě snížit.

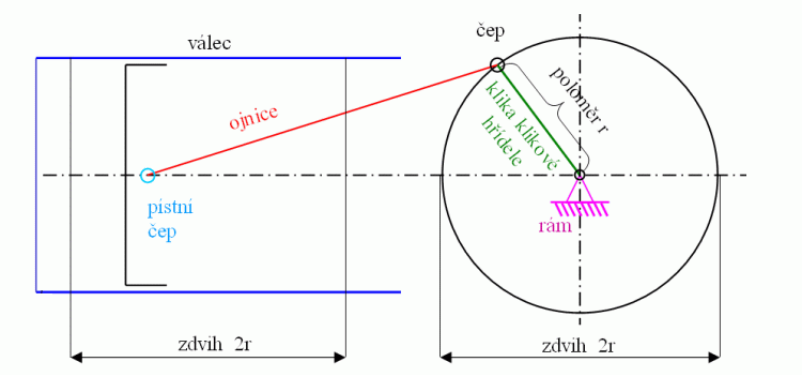
9.2. Úloha č. 2 – Klikový mechanismus

Druhou verifikační úlohou je klikový mechanismus, kde se již řeší kinematika i dynamika v závislosti na čase.

9.2.1. Zadání

Standardní klikový mechanismus (Obrázek 41) pístového spalovacího motoru je definován následujícími parametry:

- zdvih pístu (poloměr kliky): 70 (35) mm
- vzdálenost středů ok ojnice: 100 mm
- otáčky klikového hřídele: 3000 ot./min.



Obrázek 41: Schéma klikového mechanismu

Cílem úlohy je porovnání průběhů polohy, rychlosti a zrychlení pístu ve směru osy válce u analytického a numerického řešení pomocí aplikace Mechanics.

9.2.2. Analytické řešení

Analytické řešení průběhu kinematických veličin vychází z odvozených rovnic pro polohu (Rovnice 44), rychlost (Rovnice 45) a zrychlení pístu (Rovnice 46) v závislosti na úhlu natočení klikové hřídele.

$$P_{an} = r\left[1+\frac{1}{\lambda}\left(1-\sqrt{1-\sin^2\alpha}\right)-\cos\alpha\right] \tag{44}$$

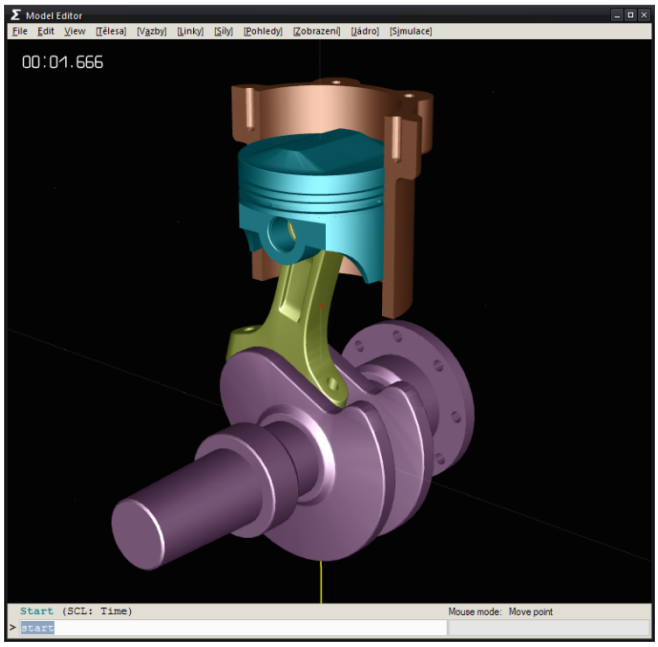
$$V_{an} = \frac{dP_{an}}{dt} = \frac{dP_{an}}{d\alpha}\cdot\frac{d\alpha}{dt} = \omega\frac{dP_{an}}{d\alpha} \tag{45}$$

$$A_{an} = \frac{dV_{an}}{dt} = \frac{dV_{an}}{d\alpha} \cdot \frac{d\alpha}{dt} = \omega \frac{dV_{an}}{d\alpha} \tag{46}$$

9.2.3. NUMERICKÉ ŘEŠENÍ

Numerické řešení bylo provedeno v aplikaci Mechanics (Obrázek 42) pro tři různé velikosti časového kroku simulace, aby bylo možné vysledovat, jaký vliv má diskretizační krok na přesnost simulace.

Volba vazebných koeficientů, které ovlivňují přesnost řešení nejvíce, byla ponechána na automatickém odhadu aplikace.



Obrázek 42: Simulace klikového mechanismu

9.2.4. SROVNÁNÍ VÝSLEDKŮ

Pro srovnání průběhů bylo zvoleno procentuální vyjádření přesnosti podle následující rovnice.

$$\Delta P_{\max} = \max\left(\left| \frac{(P_{an} - P_{nu})}{P_{an\,\max}} \right| \right) \cdot 100\% \tag{47}$$

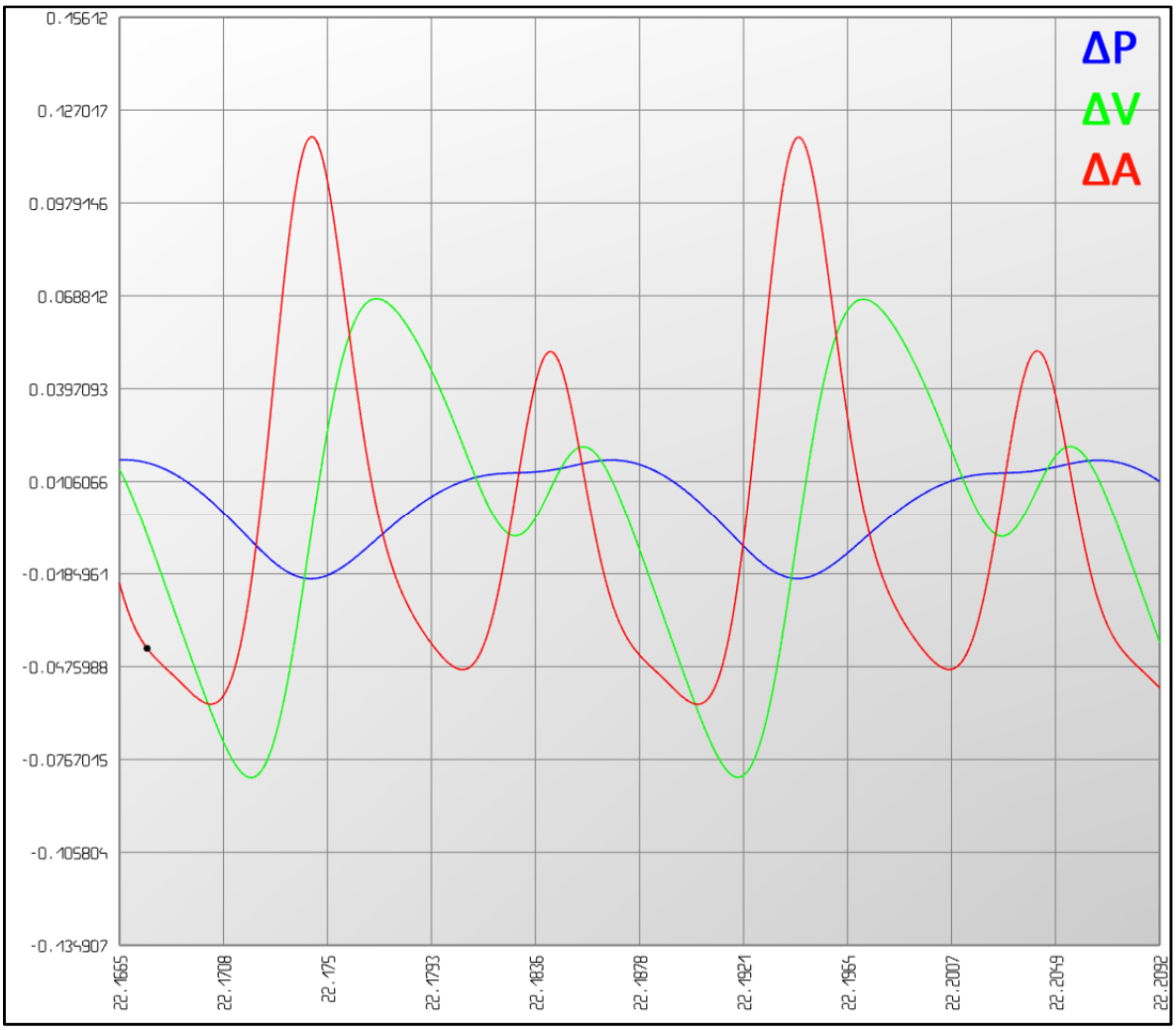
Maximální odchylka polohy pístu je vypočítána jako maximum z absolutní hodnoty poměru rozdílu numericky a analyticky získané hodnoty ku maximu analytické hodnoty. Výpočet maximální odchylky rychlosti a zrychlení pístu je zcela ekvivalentní k výpočtu maximální odchylky polohy pístu (Rovnice 47).

Následující shrnutí (Tabulka 5) srovnává maximální zjištěné procentuální odchylky kinematických veličin pístu pro tři různé diskretizační kroky simulace.

Maximální odchylka [%]		Časový krok simulace [s]		
		10^{-5}	10^{-6}	10^{-7}
Sledovaná veličina	ΔP_{max}	0.150	0.020	0.016
	ΔV_{max}	0.603	0.082	0.040
	ΔA_{max}	0.902	0.118	0.046

Tabulka 5: Přehled maximálních odchylek sledovaných veličin v závislosti na kroku simulace

Níže uvedený graf (Obrázek 43) zachycuje pro ilustraci průběh procentuálních odchylek sledovaných veličin během dvou otáček klikové hřídele při velikosti diskretizačního kroku simulace 10^{-6} sekundy (na horizontální ose je simulační čas).



Obrázek 43: Průběh přesnosti simulace při časovém kroku 10^{-6} sekundy (graf z postprocessoru aplikace Mechanics)

10. VÝKONNOSTNÍ POROVNÁNÍ

Důležitou stránkou jakéhokoliv výpočetního systému je i jeho rychlost. Proto bylo provedeno výkonnostní srovnání systému Mechanics se známým a využívaným multi-body systémem MSC ADAMS.

10.1. ZPŮSOB TESTOVÁNÍ

Testování probíhalo tak, že byl spuštěn výpočet simulace určitého modelu pro stanovenou délku simulačního času (tj. času v prostředí simulace). Doba trvání výpočtu byla změřena a z poměru trvání simulačního času ku reálnému času výpočtu se určilo výsledné skóre, udávající normalizovanou rychlost simulace. Toto skóre tedy vyjadřuje, jak rychle ubíhá simulační čas vzhledem k reálnému času (například skóre 0.5 udává, že za dvě sekundy reálného času se vypočítá jedna sekunda simulačního času).

Testování se provádělo nejprve se zapnutým překreslováním scény se simulovaným mechanismem a posléze i s vypnutým překreslováním. To proto, aby se mohlo určit, jak velký vliv na rychlost simulace vykreslování scény v jejím průběhu má.

Každé měření bylo několikrát opakováno, aby se omezila chyba měření.

10.1.1. HARDWAROVÁ KONFIGURACE

Veškeré testy probíhaly na notebooku vybaveném moderním dvoujádrovým procesorem a s následující konfigurací:

Notebook	Toshiba Tecra M5-384
Procesor	Intel Core 2 Duo T7400 (2.16 GHz)
Čipová sada	Intel i945PM Express
Operační paměť	1024 MB DDR2 (533 MHz)
Grafický akcelerátor	Nvidia Quadro NVS 110M
Pevný disk	120 GB (5400 ot./min.)

Tabulka 6: Konfigurace testovacího počítače

10.1.2. SOFTWAROVÁ KONFIGURACE

Pro testování rychlosti systému Mechanics byla použita stabilní verze z března 2008. Veškeré nastavení bylo ponecháno na výchozích hodnotách.

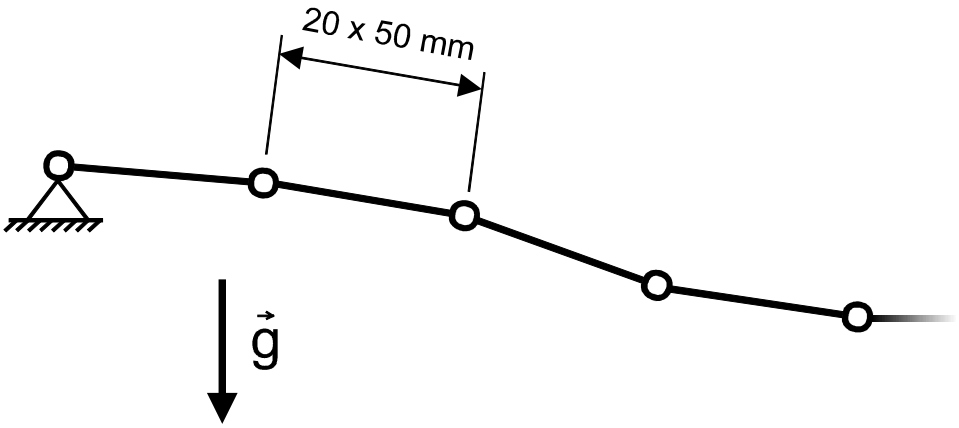
Aplikace MSC.ADAMS resp. ADAMS/View byla provozována ve verzi 2005.0.0. Nastavení aplikace bylo taktéž ponecháno na výchozích hodnotách, pouze v sekci

Dynamics byl změněn integrátor z důvodu lepší konvergence na typ Newmark a implementace nastavena na C++ .

Obě srovnávané aplikace byly provozovány na 32 bitovém operačním systému Microsoft Windows XP CZ (SP2). Ovladače grafické karty byly nainstalované ve verzi ForceWare 84.26.

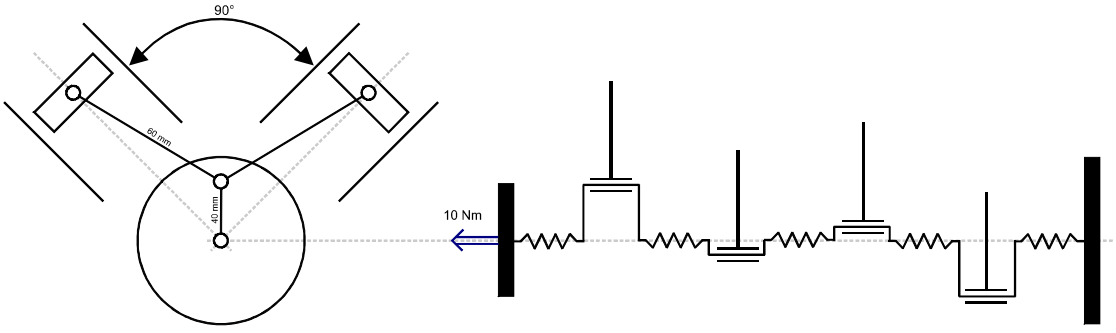
10.2. TESTOVANÉ MODELY

Jako první srovnávací mechanismus byl zvolen jednoduchý řetěz (Obrázek 44), sestávající se z dvaceti symetrických článků o délce 50 mm. Na jedné straně je řetěz přichycen sférickou vazbou k základně, druhý konec je volný. Mezi sousedními články řetězu je taktéž sférická vazba. Na řetěz působí pouze gravitační zrychlení.



Obrázek 44: Model dvacetičlánkového řetězu

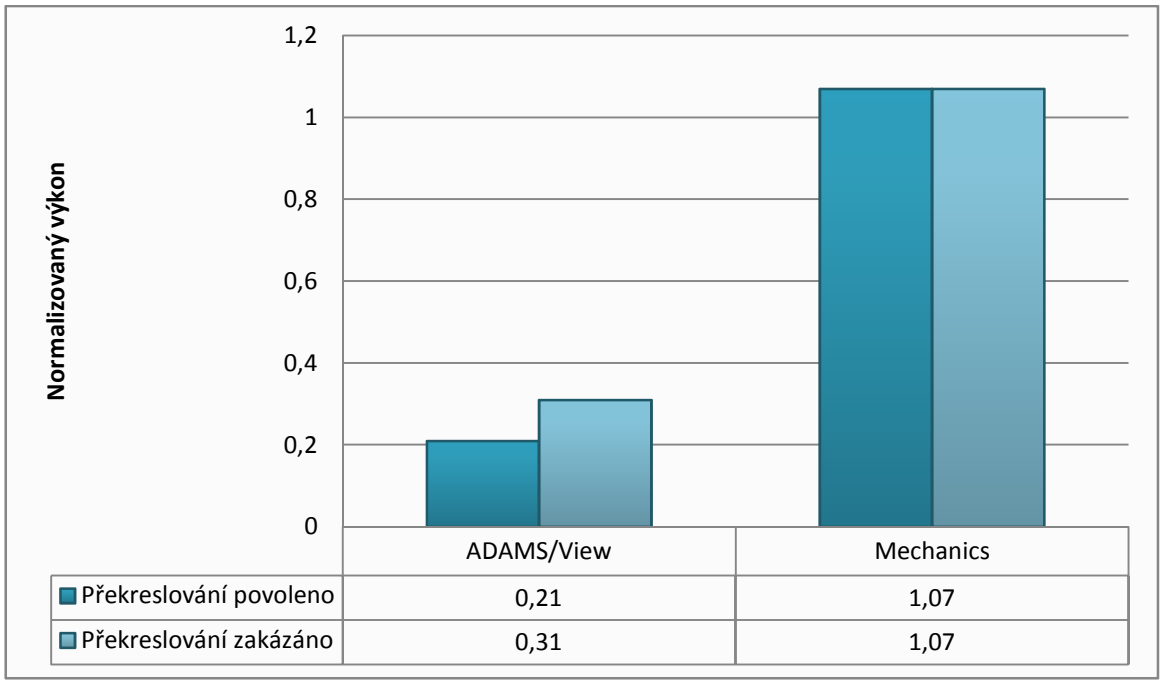
Druhým srovnávacím mechanismem je model klikového mechanismu spalovacího motoru s osmi válci uspořádanými do V (Obrázek 45). Kliková hřídel není monolitická pro celý motor, ale je složena ze šesti samostatných částí (čtyři zalomení a dva krajní setrvačníky), které jsou mezi sebou propojeny torzní pružinou (tato konfigurace by měla zároveň simulovat torzní kmitání hřídele). Na model působí opět gravitace a navíc je na předním konci hřídele zaveden kroutící moment o velikosti 10 Nm.



Obrázek 45: Model osmiválcového motoru do V

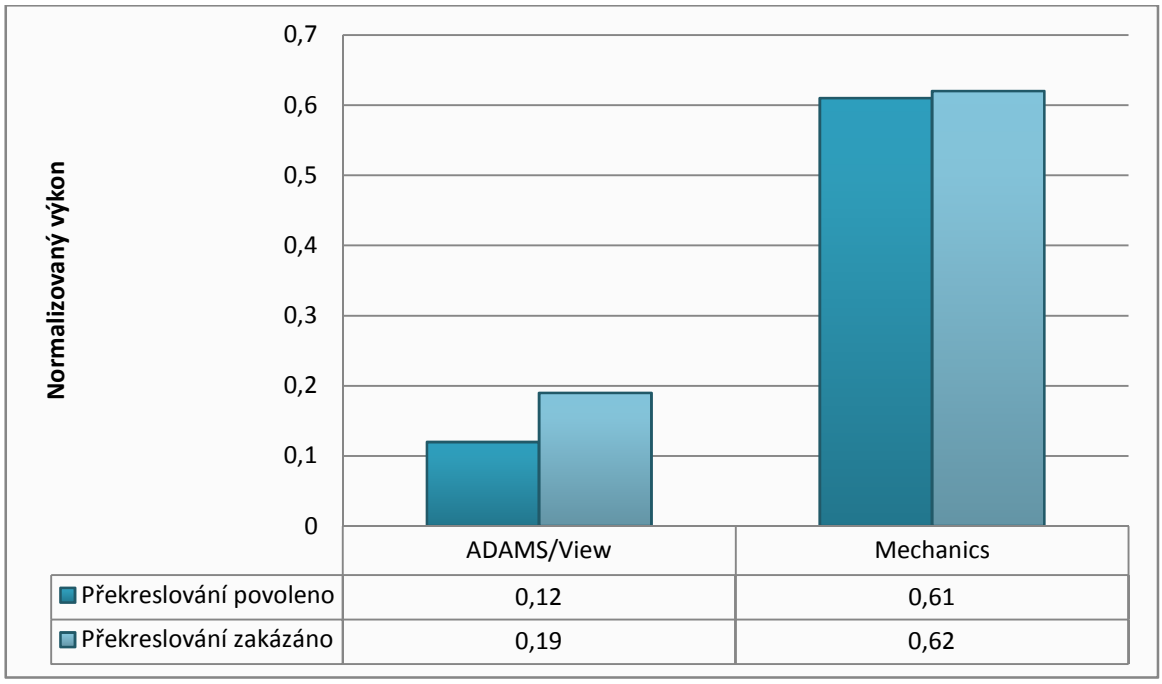
10.3. VÝSLEDKY ZÁKLADNÍCH TESTŮ

Základní testy u obou modelů probíhaly po dobu 10 sekund simulačního času. Délkou integračního kroku byla zvolena jedna milisekunda. V MKS systému Mechanics byla ovšem délka kroku nastavena na 10 mikrosekund, neboť hrubší krok vedl k nestabilitě simulace. Měření probíhalo střídavě s povoleným a zakázaným překreslováním scény.



	ADAMS/View	Mechanics
Překreslování povoleno	0,21	1,07
Překreslování zakázáno	0,31	1,07

Obrázek 46: Srovnání rychlosti simulace prvního modelu (řetěz)



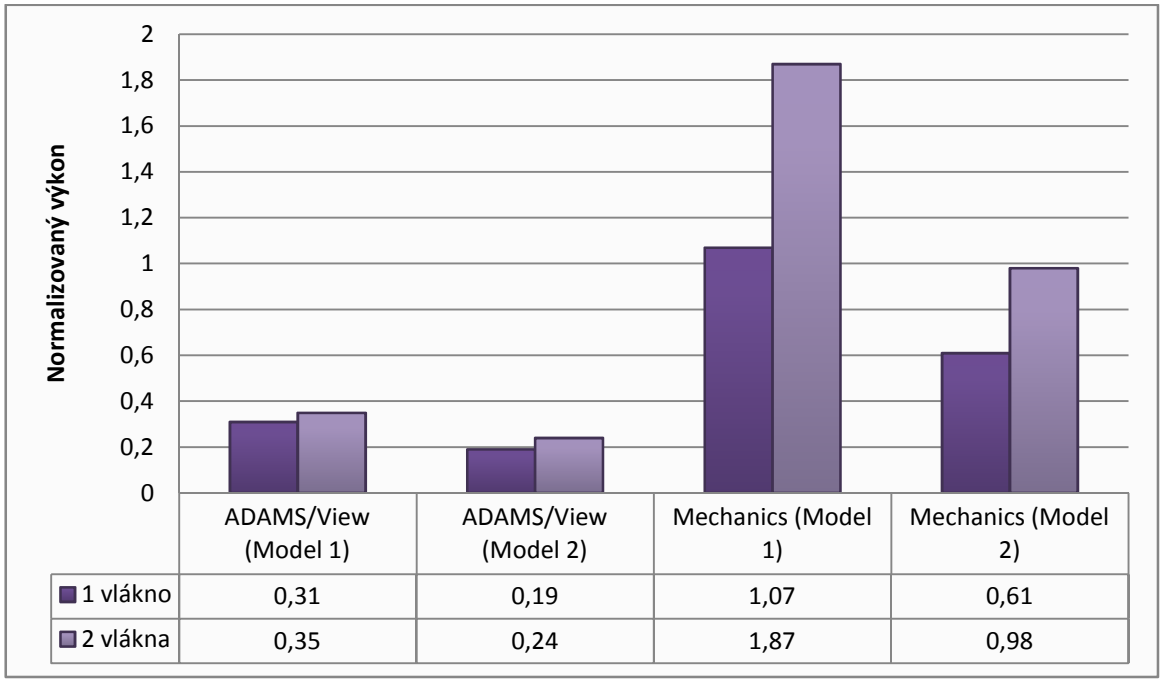
	ADAMS/View	Mechanics
Překreslování povoleno	0,12	0,61
Překreslování zakázáno	0,19	0,62

Obrázek 47: Srovnání rychlosti simulace druhého modelu (motor)

Grafy (Obrázek 46, Obrázek 47) ukazují, že v systému Mechanics probíhala simulace i přes jemnější integrační krok více než třikrát rychleji. Navíc se ukázalo, že mezi rychlostí simulace s povoleným a zakázaným překreslováním scény není na rozdíl od systému ADAMS/View prakticky žádný rozdíl. Proto může být překreslování stále zapnuto, aniž by docházelo k degradaci výkonu.

10.4. PŘÍNOS PARALELIZACE

Další měření probíhalo při povoleném využití více výpočetních vláken. Konkrétně byla v obou systémech nastavena dvě výpočetní vlákna, neboť procesor testovacího notebooku má dvě jádra. Samotné testování výkonu pak probíhalo stejně jako v předchozím případě (s vypnutým překreslováním scény u systému ADAMS/View).



Obrázek 48: Srovnání rychlosti simulace modelů s využitím jednoho a dvou výpočetních vláken

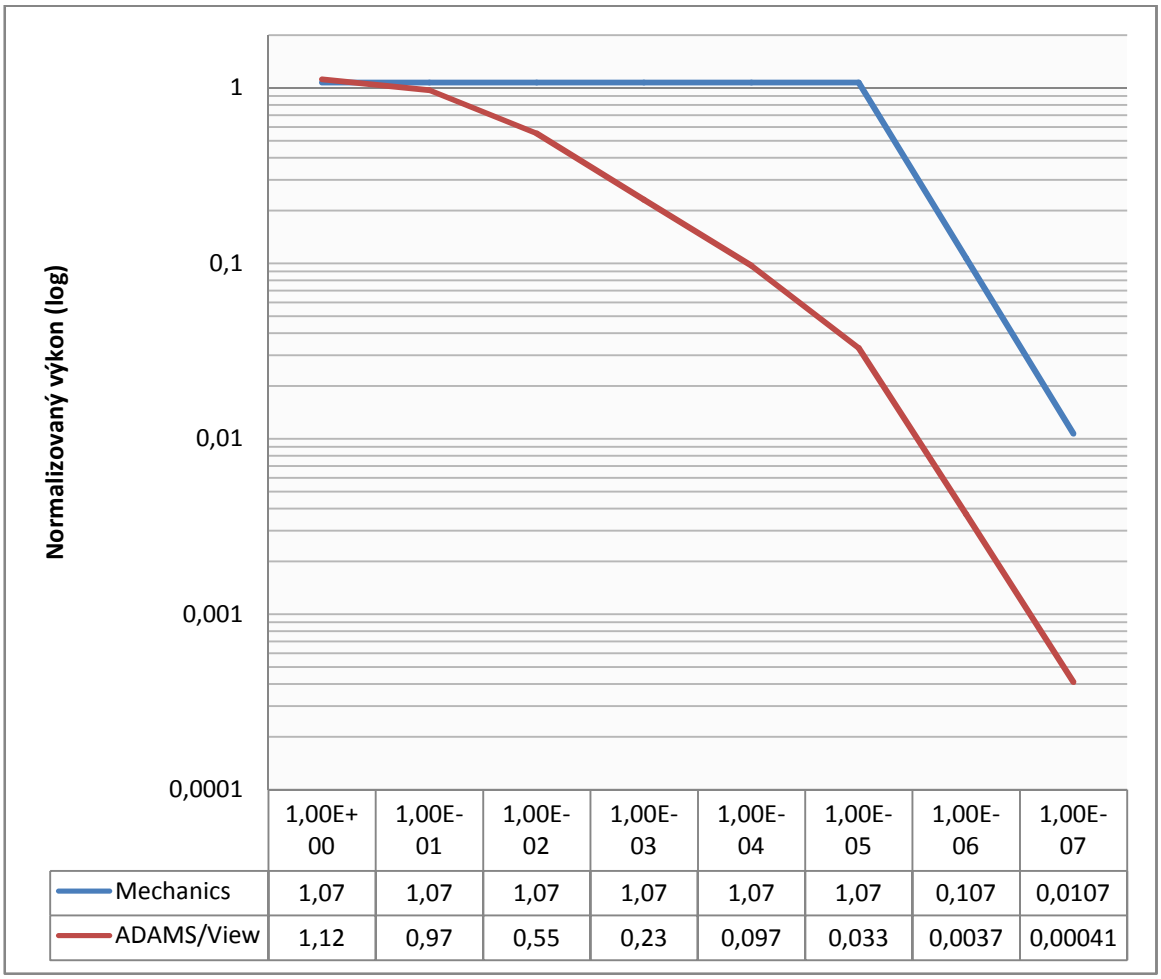
Jak je vidět z grafu (Obrázek 48), dosahuje Mechanics v obou simulovaných modelech většího nárůstu výkonu při využití dvou výpočetních vláken než ADAMS/View.

Efektivita paralelizace je v MKS systému Mechanics u modelu řetězu vyšší než u modelu motoru, přestože je řetěz složen z menšího počtu prvků, než motor. U modelu motoru je ovšem využito všech typů základních elementů, což kvůli různé délce výpočtu jednotlivých elementů zvyšuje synchronizační prostoje. Navíc musely být již ošetřeny i potenciální paměťové kolize.

10.5. VLIV INTEGRAČNÍHO KROKU

Rychlost řešení obou srovnávaných systémů se mění s tím, jak velký integrační krok je pro řešení zvolen. Proto je také důležité srovnat oba systémy při různých velikostech integračních kroků. Ke srovnání byl využit model řetězu z předchozích příkladů.

První graf (Obrázek 49) zachycuje porovnání výkonu od délky integračního kroku jedna sekunda až po délku desetiny mikrosekundy. Logaritmické vyjádření osy Y v grafu bylo zvoleno z důvodu velkých výkonnostních rozdílů mezi oběma srovnávanými aplikacemi. K testování byl využit model řetězu z předchozích příkladů.



	1,00E+00	1,00E-01	1,00E-02	1,00E-03	1,00E-04	1,00E-05	1,00E-06	1,00E-07
Mechanics	1,07	1,07	1,07	1,07	1,07	1,07	0,107	0,0107
ADAMS/View	1,12	0,97	0,55	0,23	0,097	0,033	0,0037	0,00041

Obrázek 49: Srovnání výkonu v závislosti na integračním kroku

Počáteční konstantní výkon aplikace Mechanics je dán tím, že integrační krok metody MKS musí být velmi jemný. Proto i pro větší intervaly vzorkování měřených dat je stále nastaven integrační krok na deset mikrosekund.

I přes tuto skutečnost je vidět, že systém ADAMS/View dosáhl vyšší rychlosti simulace až při integračním kroku o délce jedné sekundy. Za povšimnutí také stojí fakt, že rychlost simulace v závislosti na délce kroku neklesá u tohoto systému lineárně.

10.6. POROVNÁNÍ HW PLATFOREM

Důležité pro komplexní porovnání výkonnosti výpočetních aplikací je také ověření toho, jak se tyto aplikace chovají na různých HW platformách, které mohou být v dnešní době poměrně rozmanité. Vzhledem k tomu, že aplikace Mechanics je psána pro běh pod operačním systémem Microsoft Windows, bylo testování omezeno na procesory x86 kompatibilní výrobců Advanced Micro Devices (AMD) a Intel.

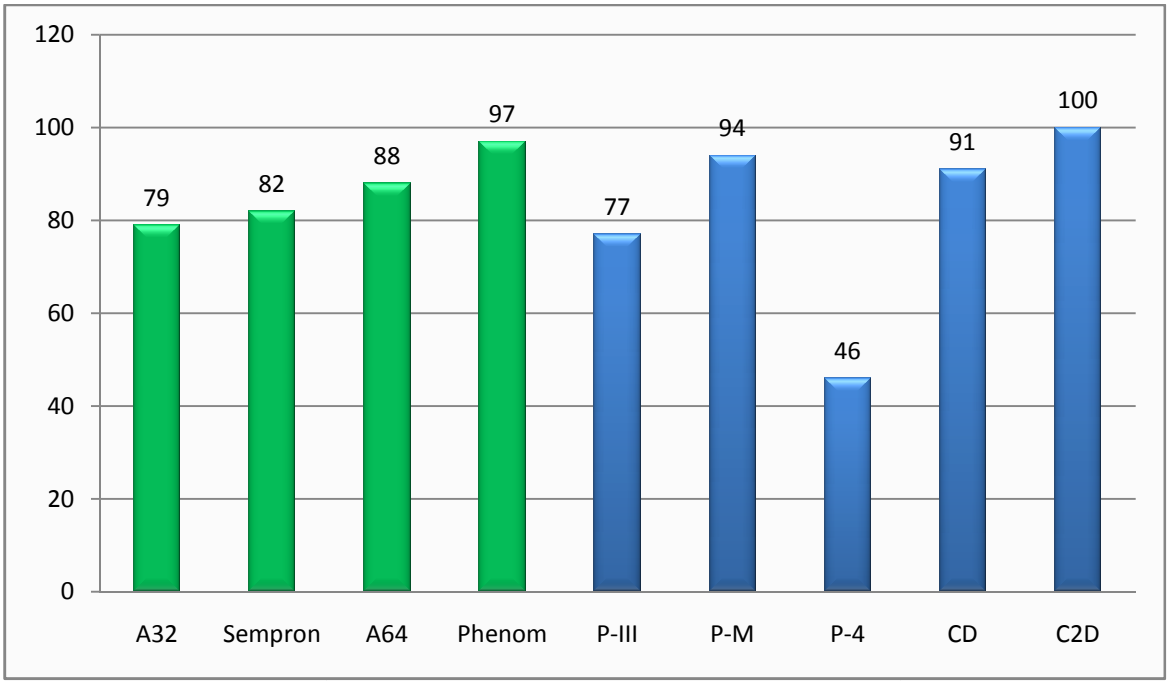
Výkon byl měřen přímo aplikací Mechanics a to sice jako počet iterací simulace, které proběhnou během jedné sekundy (jednotka kS – sto tisíc iterací za sekundu). Aby se minimalizoval vliv různých grafických karet testovaných systémů, bylo vykreslování během simulace zakázáno, přestože ve většině případů nemá na měřený výkon prakticky vliv.

Přestože některé testované procesory byly dvoujádrové, byl test prováděn v základním jednovláknovém režimu, aby se lépe projevil vliv samotné architektury jádra a zejména výpočetní výkon jednotek pro práci s reálnými čísly (FPU jednotky).

Z měření (Tabulka 7, Obrázek 50) vyplývá, že v současné době nejpoužívanější generace x86 kompatibilních procesorů obou konkurenčních výrobců jsou z hlediska výkonu v aplikaci Mechanics víceméně srovnatelné. Jedinou výjimkou je architektura NetBurst použitá u procesoru Intel Pentium 4. Tento procesor sice díky dlouhé pipeline může dosahovat poměrně vysokých frekvencí, nicméně jeho FPU jednotka (pro instrukce x87) je natolik slabá, že normalizovaný ani absolutní výkon zdaleka nestačí na jakoukoliv konkurenci.

Výrobce	Procesor	Jádro	Frekvence [GHz]	Naměřený výkon [kS]	Výkon na 1 GHz [kS]	Relativní porovnání [%]
AMD	Athlon (32)	T.Bred-B	2,1	65,2	31,1	79
	Sempron	Palermo	1,6	51,3	32,1	82
	Athlon 64	-	1,8	62,2	34,6	88
	Phenom	-	2,4	91,2	38,0	97
Intel	Pentium III	-	1,0	30,1	30,1	77
	Pentium M	-	2,0	73,3	36,7	94
	Pentium 4	Northwood	3,0	53,6	17,9	46
	Core Duo	-	1,8	63,9	35,5	91
	Core 2 Duo	**Conroe**	**2,6**	**101,7**	**39,1**	**100**

Tabulka 7: Výsledky měření výkonu na různých platformách



Obrázek 50: Srovnání normalizovaného výkonu jednotlivých procesorů (100 % vztaženo k platformě s procesorem Core 2 Duo)

10.7. SHRNUTÍ

Systém Mechanics dokázal testovací modely simulovat výrazně vyšší rychlostí, přestože integrační krok byl zvolen o dva řády menší a tudíž i výsledky jemnější. Navíc na rozdíl od systému ADAMS/View nezpůsobuje povolené překreslování scény během simulace prakticky žádný propad výkonu.

Taktéž efektivnost paralelizace (čili výkonnostní nárůst při využití víceprocesorových systémů) je větší, což dále zvyšuje výkonnostní rozdíl obou srovnávaných aplikací na víceprocesorových systémech.

Zároveň je však třeba říci, že poměr rychlosti obou systémů záleží na tom, jak velký integrační krok je od řešení modelu požadován. MKS totiž vyžaduje velmi jemný integrační krok (například u testovaných mechanismů nebylo možné použít větší krok než 10^{-5}, jinak docházelo k nestabilitě řešení). Pokud tedy stačí hrubé řešení s velkým časovým krokem, může být systém ADAMS/View rychlejší.

Zjednodušeně by se dalo říci, že zatímco v multi-body systému ADAMS rychlost řešení kolísá s tím, jak se mění integrační krok pro dosažení předvolené přesnosti, tak v systému Mechanics se udržuje konstantní integrační krok (a tím i rychlost simulace) a v závislosti na podmínkách se pak mění přesnost řešení.

11. ZÁVĚR

11.1. DOSAŽENÉ CÍLE

Cílem této diplomové práce bylo seznámení čtenáře se základními principy a možnostmi multi-body systému Mechanics. Během vývoje aplikace se podařilo dosáhnout zajímavých parametrů zejména z hlediska rychlosti simulace, kdy i na běžném počítači je možno simulovat v reálném čase mechanismy složené až z několika desítek těles. Nemalá pozornost byla věnována také možnostem grafického prostředí, interaktivity simulace a tématu virtuální reality, které autora této práce v poslední době velmi zaujalo. Vzhledem k rozsahu vytvořené aplikace, jenž dokumentují níže uvedené údaje, však nemohly být popsány zdaleka všechny její části a schopnosti.

- 3 roky vývoje
- přes 60 000 řádek originálního kódu

11.2. VÝHODY A NEVÝHODY

Následující seznam shrnuje v bodové podobě základní kladné i záporné vlastnosti systému Mechanics, které odpovídají aktuální vývojové verzi.

\+ Vysoká rychlost simulace

\+ Interaktivní prostředí a simulace, včetně podpory virtuální reality

\+ Integrovaný postprocessing

\+ Nízké systémové nároky

\+ Podpora moderních technologií

\+ Modulární koncepce a programovatelnost

± Velmi malý diskretizační krok simulace

– Neschopnost řešit čistě kinematické úlohy bez vlivu dynamiky

– V současné podobě nabízí pouze základní kinematické vazby

– Občasná nestabilita aplikace

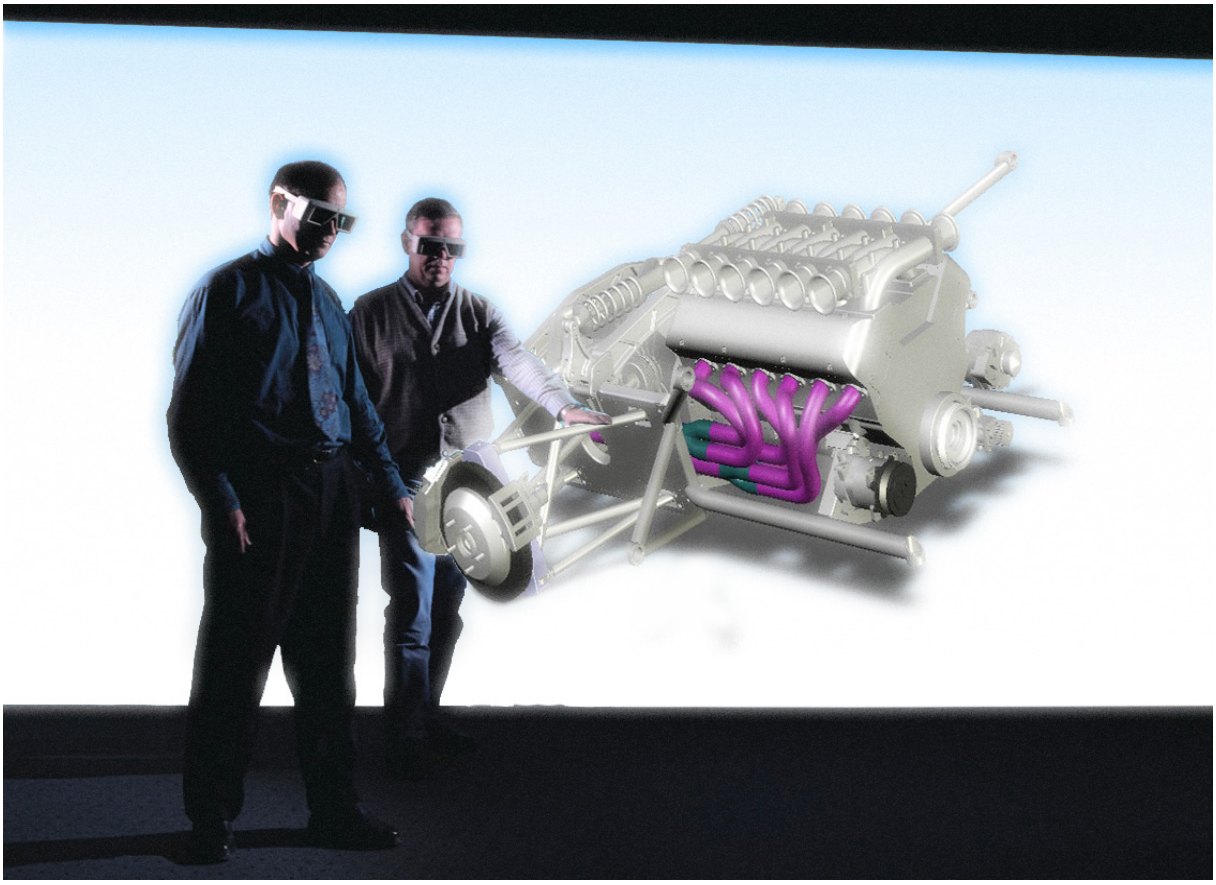
11.3. SMĚR DALŠÍHO VÝVOJE

Přestože celkový směr vývoje aplikace není možné předem zaručeně odhadnout, tak prioritu vývojových prací na multi-body systému Mechanics mají v aktuální fázi vývoje zejména následující tři oblasti.

- **Rychlost** – výpočetní jádro programu neustále prochází úpravami za účelem dosažení co možná nejlepších výkonových parametrů. Hlavním zájmem je optimalizace paralelního zpracování výpočtu pro víceprocesorové, resp. vícejádrové počítače.
- **Interaktivita a grafické prostředí** – velký důraz je při vývoji kladen také stále na posilování grafických schopností a interaktivnosti simulačního prostředí v návaznosti na prostředky virtuální reality.
- **Numerické optimalizace** – další hlavní oblastí vývoje je i snaha o zlepšování kvalitativních parametrů simulace, tj. přesnosti a numerické stability, při současném zachování vysoké rychlosti řešení.

Kromě výše uvedených bodů se vývoj dále zaměřuje také na možnosti vzájemného propojení systému Mechanics s dalšími vhodnými aplikacemi (např. Matlab), stejně jako na soustavné rozšiřování podpůrných funkcí programu v oblastech tvorby modelu, řízení simulace a následné zpracování získaných dat.

Závěrečný obrázek této práce ilustruje vývojovou vizi systému Mechanics, kdy by sestavování složitého modelu stejně jako jeho simulace a přímé ovládání uživatelem probíhaly v reálném čase v prostředí virtuální reality.



Obrázek 51: Vize budoucna...

POUŽITÉ SYMBOLY A ZKRATKY

V rovnicích a matematických vztazích užitých v této diplomové práci jsou použity symboly v kombinaci s indexy dle následujícího přehledu.

Symboly

e	*[1]*	*jednotkový vektor*
F	*[N]*	*síla*
M	*[Nm]*	*moment*
m	*[kg]*	*hmotnost tělesa*
I	*[kg.m²]*	*tenzor setrvačnosti*
L	*[m]*	*obecná vzdálenost*
g	*[m.s^{-2}]*	*gravitační zrychlení*
P	*[m]*	*poloha*
V	*[m.s^{-1}]*	*rychlost*
A	*[m.s^{-2}]*	*zrychlení*
t	*[s]*	*čas*
C	*[1]*	*transformační matice pro převod souřadných systémů*
Q	*[1]*	*transformační matice pro výpočet rotační polohy*
K	*[1]*	*matice vazebných koeficientů*
H	*[N.m^{-1}]*	*součinitel tuhosti pružiny*
D	*[N/m.s^{-1}]*	*součinitel tlumení pružiny*
u	*[1]*	*míra přenosu tlakových sil*
v	*[1]*	*míra přenosu tahových sil*
r	*[m]*	*délka ramene klikové hřídele*
λ	*[1]*	*klikový poměr*
α	*[rad]*	*úhel natočení klikové hřídele*
ω	*[rad.s^{-1}]*	*úhlová rychlost klikové hřídele*

Indexy symbolů

cg	*těžiště*
i	*i-tý bod tělesa (nula pro těžiště)*
t	*translační*
r	*rotační*
a	*doplňková*
tr	*translační relativní (vůči těžišti tělesa)*
rr	*rotační relativní (vůči těžišti tělesa)*
d	*diferenční*

K	*korekční*
k-1	*předchozí iterace*
A	*první vázaný bod*
B	*druhý vázaný bod*
P	*bod působiště*
S1	*první sloupec matice*
S2	*druhý sloupec matice*
S3	*třetí sloupec matice*
an	*analytický*
nu	*numerický*
min	*minimální*
max	*maximální*
red	*redukovaná/ý*
?	*variabilní index*

Zkratky

MKS	*metoda korekčních sil*
GSS	*globální souřadný systém*
LSS	*lokální souřadný systém*
CPU	*centrální mikroprocesor počítače*
FPU	*jednotka mikroprocesoru pro práci s reálnými čísly*
SIMD	*označení pro skupinu vektorových instrukcí*
SSE2	*SIMD instrukce 2. generace firmy Intel*

LITERATURA

[1] PORTEŠ, P. Matemaitcké modelování ovladatelnosti vozidel. Brno: VUT Brno, 199x

[2] KREITH, F. The CRC Handbook of Mechanical Engineering. 1. vyd. 1998. ISBN 978-0849394188.

[3] MSC Software Corporation [online]. Dostupné z: <http://www.mscsoftware.com>

[4] Wikipedie, otevřená encyklopedie [online]. Dostupné z: <http://wikipedia.org/>

[5] Builder – Informační server o programování [online]. Dostupné z: <http://builder.cz/>

[6] GALI-3D: Technologie 3D a stereoskopie [online]. Dostupné z: <http://www.gali-3d.com/cz/techno/techno.php>

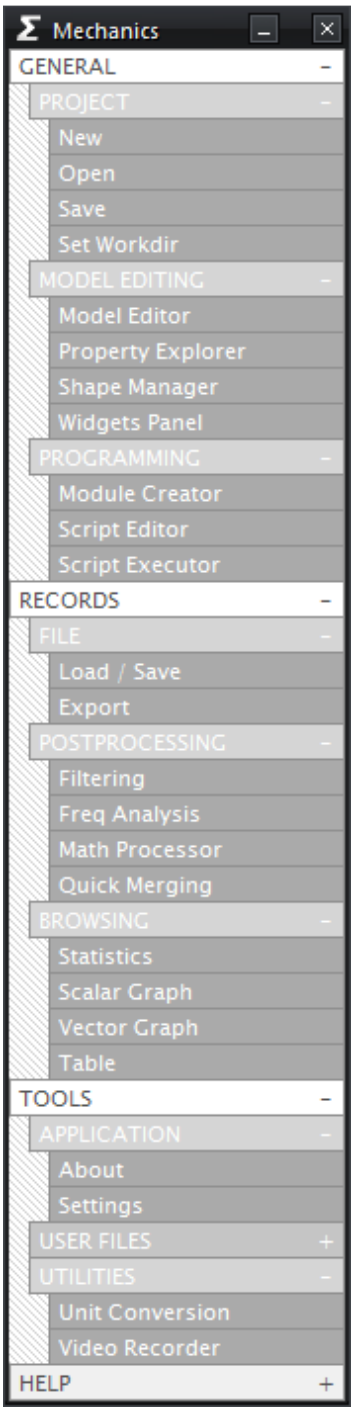
[7] Intel® 64 and IA-32 Architectures Software Developer's Manuals [online]. Dostupné z: <http://www.intel.com/products/processor/manuals/index.htm>

[8] AMD – Technical Documentation [online]. Dostupné z: <http://www.amd.com/gb-uk/Processors/TechnicalResources/>

[9] OpenGL – The Industry Standard for High Performance Graphics [online]. Dostupné z: <http://www.opengl.org>

PŘÍLOHY

PŘÍLOHA Č. 1 – HLAVNÍ NÁSTROJE GUI



- ➔ Založení nového projektu
- ➔ Otevření souboru s modelem
- ➔ Uložení aktuálního modelu do souboru
- ➔ Nastavení pracovního adresáře

- ➔ Hlavní okno s editorem modelů
- ➔ Správce parametrického stromu
- ➔ Správce knihovny 3D tvarů
- ➔ Panel interaktivních ovládacích prvků

- ➔ Průvodce programováním rozšiřujících modulů
- ➔ Editor skritpů konzole
- ➔ Nástroj pro rychlé spouštění parametrizovaných skriptů

- ➔ Načítání a ukládání hotových záznamů ze simulace
- ➔ Export dat záznamů do univerzálního formátu CSV

- ➔ Matematické filtrování zaznamenaných dat
- ➔ Frekvenční analýza zaznamenaných dat
- ➔ Obecný matematický procesor pro práci se záznamy
- ➔ Nástroj pro spojování záznamů stejného typu

- ➔ Statistická analýza záznamů
- ➔ Prohlížení dat ve vícekanálovém skalárním grafu
- ➔ Prohlížení dat ve vektorovém grafu
- ➔ Prohlížení výpisu dat v tabulce

- ➔ Informace o aplikaci
- ➔ Základní nastavení aplikace
- ➔ Editace uživatelských souborů aplikace

- ➔ Rychlá konverze jednotek
- ➔ Nástroj pro pořizování videozáznamů ze simulace
- ➔ Nápověda k aplikaci

PŘÍLOHA Č. 2 – SEZNAM PŘÍKAZŮ KONZOLE

Ve špičatých závorkách **<>** jsou uvedeny povinné parametry, v kulatých závorkách **()** nepovinné parametry a ve složených závorkách **{}** jsou parametry výčtového typu, tj. jako parametr se uvede vždy jeden prvek z nabízeného seznamu.

Add <REF: Node>
Přidá element do parametrického stromu. Tímto příkazem lze přidávat všechny základní elementy mechanismu a navíc také simulační záznamy a rozšiřující moduly.

V případě přidávání rozšiřujícího modulu má příkaz ještě jeden parametr, který určuje jméno souboru DLL knihovny modulu. Toto jméno se uvádí bez koncovky *.dll.

Asm
Spustí speciální mód simulace určený ke složení těles mechanismu do funkční polohy odpovídající kinematickým vazbám mezi tělesy.

Bench <SCL: Time>
Spustí simulaci na dobu určenou parametrem Time a po dokončení vrátí průměrnou výkonnost simulace ve stotisících iterací za sekundu.

CInfo
Pomocný příkaz pro zjištění informací o jádře. Příkaz nemá žádné parametry.

Clear {hist, states, rdone}
Vymaže jeden z uvedených zásobníků (hist - historie příkazů konzole, states - uložené stavy simulace, rdone - pořízené záznamy simulace).

Del <REF: Node>
Smaže uvedený element parametrického stromu. Příkaz lze použít na všechny základní elementy modelu a dále na moduly a záznamy.

Dis <VEC: Range>
Rozmístí tělesa vůči jejich aktuální poloze náhodně v prostoru určeném poloměrem, který je předán jako vektorový parametr.

Inc <REF: ScalarNode> <SCL: Increment>
Přičte k uvedené skalární proměnné v prvním parametru hodnotu uvedenou v druhém parametru.

Mul <REF: ScalarNode> <SCL: Factor>
Vynásobí skalární proměnnou uvedenou v prvním parametru hodnotou uvedenou v druhém parametru.

Enum <REF: Node>
Vypíše seznam všech poduzlů uvedeného uzlu parametrického stromu.

Get <REF: Node>
Vrátí aktuální hodnotu reference uvedené v parametru příkazu.

List {hist, states, records, modules}
Vypíše jeden z uvedených zásobníků.

PaintS {bodies, binds, links, forces}
Obarví prvky uvedené skupiny elementů postupně barvami spektra.

PaintR {bodies, binds, links, forces}
Obarví prvky uvedené skupiny elementů náhodnými barvami.

Pop <STR: StateName>
Vyzvedne z paměti dočasně uložený stav simulace a nahradí jím aktuálně otevřený model.

Push <STR: StateName>
Uloží aktuální stav simulace do paměti pod uvedeným názvem. Takto uložený stav je možné kdykoliv později okamžitě vyzvednout. Po ukončení programu se ovšem všechny tyto dočasné stavy smažou.

Relax
Odebere tělesům a vazbám veškerou kinetickou energii (tj. vynuluje vektory rychlostí a zrychlení). Příkaz nemá žádné parametry.

Ren <REF: Node> <STR: NewName>
Přejmenuje uzel parametrického stromu. První parametr tvoří kompletní referenci na přejmenovávaný uzel, druhý parametr pak obsahuje pouze nové jméno uzlu. Tento příkaz lze aplikovat na všechny základní elementy mechanismu.

Set <REF: Node> <VAR: Value>
Nastaví hodnotu prvku parametrického stromu. První parametr určuje referenci na nastavovaný prvek, druhý parametr je tvořen zápisem nastavované hodnoty v příslušném tvaru.

Start (SCL: Time)
Spustí simulaci na dobu uvedenou v parametru Time. Pokud žádný parametr uvedený není, poběží simulace tak dlouho, dokud nebude zastavena uživatelem.

Stat
Vypíše stručnou statistiku modelu (počty elementů jednotlivých skupin a odhady energií).

Stop
Zastaví běžící simulaci.

PŘÍLOHA Č. 3 – UKÁZKOVÝ SKRIPT KONZOLE

```
// název programu (nemá pro skript vliv)
program Script;

// značka pro vložení pomocných rutin do kódu
<include_aux>

// klíčové slovo definující začátek procedury (v tomto případě
hlavního bloku programu)
Begin

  // přímý příkaz konzole - vložení tělesa
  Cmd('add bodies.pokus');

  // přímý příkaz konzole - nastavení pozice tělesa
  Cmd('set bodies.pokus.center.pt 0;0.5;0');

  // zapíše do výstupu konzole určený text (vždy na nový řádek)
  WriteOut('Tento text se v konzoli neobjeví.');

  // smazání výstupu konzole
  ClrRes;

  // nastavení hodnoty vektoru po složkách
  SetVector3('core.gravity', 0, -20, 0);

  // nastavení hodnoty skaláru
  SetScalar('core.dt', 1e-7);

  // zapíše do výstupu konzole určený text
  WriteOut('Změněny parametry:');

  // vložení prázdného řádku do výstupu konzole
  WriteOut('');

  // zapíše do výstupu konzole hodnoty proměnných
  WriteOut('gravitace v ose Y =
    '+FloatToStr(GetScalar('core.gravity.y')));
  WriteOut('diskretizační krok =
    '+FloatToStr(GetScalar('core.dt')));

// klíčové slovo definující konec procedury (v tomto případě
hlavního bloku programu)
end.
```