Българска Академия на Науките

Институт за Космически Изследвания и Технологии

секция „Атмосферни оптични изследвания“

Атанас Маринов Атанасов

Разработка на методи, алгоритми и програмни средства за анализ и проектиране на космически мисии

автореферат

на дисертация за присъждане на образователна и научна степен

ДОКТОР

Област на висше образование: 5 Технически науки
Професионално направление: 5.5 Транспорт, корабоплаване и авиация
Научна специалност: „Динамика, балистика и управление на полета на летателни апарати“

Научен консултант: Проф. дтн Явор Георгиев Чапанов

София 2019

Дисертационния труд е обсъден и предложен за защита на заседание на разширен семинар на секция „Атмосферни оптични изследвания“ при **ИКИТ – БАН**, филиал в г. Стара Загора.

Докторантът работи като асистент в секция „Атмосферни оптични изследвания“, филиал на ИКИТ в г. Стара Загора – БАН.

Данни за дисертационния труд

Дисертационният труд съдържа пет глави изложени на 132 страници. Включва 32 фигури, 1 таблица, 21 илюстративни програмни фрагментра, 224 литературни източници и 13 научни публикации чиито резултати са отразени в дисертацията. Към основния текст са добавени основни подпрограми, разпределени в поредица допълнения **A**, **B**, **C**, **D** и **E**.

Научно жури

Външни членове:

1. проф. дтн инж. Явор Георгиев Чапанов, НИГГГ – БАН
2. доц. д-р инж. Кирил Методиев Алексиев, ИИКТ – БАН
3. доц. д-р Пенчо Генов Маринов, ИИКТ – БАН

Вътрешни членове:

4. доц. д-р инж. Дойно Иванов Петков – ИКИТ – БАН
5проф. д-р инж. Румен Дончев Недков – ИКИТ – БАН

Резервни членове:

1. доц. д-р инж. Любомир Николов Бончев, пенсионер
2. проф. дтн инж. Георги Ставрев Сотиров, ИКИТ – БАН

Защитата на дисертационната работа ще се състои на . .2019 от ч. в зала на Института за космически изследвания и технологии – БАН, ул. „Акад. Георги Бончев“, бл. 1.

Материалите по защитата са на разположение на интересуващите се в канцеларията на Института за космически изследвания и технологии – БАН, ул. „Акад. Георги Бончев“, бл. 1.

Съдържание

Списък на използвани съкращения

АПИСОДУ	**-** **А**ктуален **П**аралелен **И**нтегратор на **С**истеми от **О**бикновени **Д**иференциални **У**равнения
ИН	**-** **И**зчислителна **Н**ишка
ОДУ	- **О**бикновени **Д**иференциални **У**равнения
ОПН	**-** **О**бединение на **П**улове от **Н**ишки
ПИСОДУ	- **П**аралелен **И**нтегратор на **С**истеми от **О**бикновени **Д**иференциални **У**равнения
ППСА	- **П**аралелен **П**роцесор за **С**итуационен **А**нализ
ПСА	- **П**роцесор за **С**итуационен **А**нализ
ПСО	- **П**ланиране на **С**пътникови **О**перации
СА	- **С**итуационен **А**нализ
СОДУ	- **С**истеми от **О**бикновени **Д**иференциални **У**равнения
СО	- **С**пътникови **О**перации
ПССКМЕ	- **П**рограмна **С**истема за **С**имулации на **К**осмически **М**исии и **Е**ксперименти

Обща характеристика на дисертационния труд

Актуалност и значимост на темата

Актуалността на темата произтича от мястото и ролята на средствата за компютърни симулации при анализа на космически мисии и експерименти на различните етапи от подготовката и осъществяването им. Fortescu (2011) разглежда различни етапи от подготовката и осъществяването на космически мисии, като поставя акцента върху инженерни аспекти. Други автори (Wertz & Larson, 1999; Montenbruck & Gill, 2001; Vallado, 2013) наблягат върху динамиката на обектите и геометрични аспекти, свързани с условия и ограничения при провеждане на експериментите, а мястото на математическите методи за моделиране и разработката на средства за компютърни симулации при разработката на космически мисии е разгледано от Rainey (2004) и Eickhoff (2009).

В последно време все повече държави и организации се включват в реализацията на космически проекти, като използването на мини, микро-, нано спътникови технологии изключително разширява възможностите за това (OneWeb, Multi-satellite projects). С разширяване на тези възможности се увеличава и необходимостта от изследвания и разработки на универсални средства за проектиране, които могат да бъдат прилагани към широк клас от спътникови мисии и експерименти. Вниманието се насочва към реализация на проекти с голяма размерност, свързана с брой на обекти, апаратурата и задачите за решаване (OneWeb, Multi-satellite projects). Увеличаването на възможностите за извеждане на космическа апаратура в космоса води също до нарастване на опасността и риска от стълкновения между отделните активни и неактивни обекти и налага развитие на средства за оценка и намаляване на риска, както на етапа на проектиране, така и в оперативен режим.

От друга страна, навлизането на нови технологии в областта на процесорите и тяхното обкръжение предлага възможности за много по-бързи изчисления и за прилагане на по-сложни и точни физични модели. Използването на такива модели изисква разработката и изследването на паралелни алгоритми за тяхната реализация. Използването на нарасналата изчислителна мощ изисква задълбочен анализ и изследване на различни методи, алгоритми и изчислителни модели, за да бъдат прилагани в областта на космическите изследвания.

В областта на ситуационния анализ се установява дефицит от специфични методи и алгоритми, които да го представят като самостоятелна дисциплина. Представен е формализъм както по отношение на представянето на ситуационните задачи, така и по отношение на тяхното решаване. Тази формализация издига на по-високо, абстрактно ниво ситуационния анализ и позволява развитието на нови формални методи за решаване и оптимизация на алгоритмите.

Разработените средства за интегриране на уравненията на движение на спътници и извършване на ситуационен анализ са приложими при симулации на системи с големи размерности. С нарастването на населеността на околоземното пространство, при възникване на риск от стълкновение на спътници в оперативен режим, ще се налага използването на средства за справянето с проблема.

Представените алгоритми могат да бъдат използвани за решаването на такива задачи.

Цел и задачи на изследването

Целта на дисертацията е да се разработят методи, алгоритми и компютърни програми, приложими при проектиране и анализ на космически мисии и експерименти. Една по-далечна цел е разработка на компютърна среда за симулации на активни (физически) спътникови експерименти в космоса.

За постигане на целта са решени следните задачи:

1). Разработка на стратегия за избор на оптимална схема за числено интегриране; анализ на изчислителната ефективност, при прилагане на стратегията за избор на оптимална изчислителна схема, и устойчивостта на решението; разработка на сериен вариант на интегратор за решаване на системи диференциални уравнения.

2). Разработка на паралелен интегратор на системи от диференциални уравнения и изследване на ефективността му за интегриране на уравненията на движение на голям брой обекти. Разработка на модел за представяне на смущения.

3). Разработка на модел за представяне на ситуационни задачи и сериен вариант на процесор за ситуационен анализ (**ПСА**).

4). Разработка на паралелен вариант на процесор за решаване на ситуационни задачи и изследване на ефективността от прилагането му, при решаване на задачи с голяма размерност.

5). Разработка на програмен модел за паралелни изчисления – „обединение на пул от нишки“.

6). Разработка на алгоритми за планиране на операции и прилагането им при планиране на „свръх-заявени” спътникови операции.

7). Разработка на програмна система за симулации на космически мисии.

8). Извършване на експеримент по прилагане на разработваната система .

Обем и структура на дисертацията

В първа глава се посочва от какво произтича актуалността на темата. Във втора глава е направен интердисциплинарен обзор по темата на дисертацията. В трета глава са посочени цел и задачи за решаване. В четвърта глава са изложени резултати от извършената работа, свързани с разработката на изчислителни инструменти. Пета глава предлага кратко приложение на разработените изчислителни инструменти в рамките на система за симулации на космически експерименти. В шеста глава освен заключение се посочват и възможности за бъдеща работа. Приносите на дисертационния труд са изброени в седма част, а в осма е изложен списък с публикациите на автора по темата на дисертацията. След цитираната литература е приложено допълнение, съдържащо текста на подпрограми по представените изчислителни инструменти.

Библиографията съдържа 225 източника.

Научна новост:

- Интегратор на системи от обикновени диференциални уравнения се основава на стратегия за избор на оптимална схема;
- Модели за представяне на ситуационни условия и ситуационни задачи;
- Евристики за оптимизация на ситуационния анализ;
- Разработен е вариант на програмния модел „пул от изчислителни нишки“, приложим към различни изчислителни инструменти;
- Програмен модел „обединение на пулове от нишки“ за съвместно използване на ресурсите на системата от паралелни изчислителни инструменти.

Практическа приложимост

Разработените изчислителни инструменти са включени в разработвана система за анализ и проектиране на космически мисии. Те са приложими и в други системи според предназначението си. Приложен е пример и резултати от приложение на системата.

Основно съдържание на дисертационния труд

Глава I. Литературен обзор

Извършен е интердисциплинарен обзор, основан на 225 литературни източника.

1.1. Ролята на симулациите за космическите изследвания

Различни аспекти, свързани със симулация на динамиката и геометрията на космически мисии като цяло, са изложени подробно от Wertz & Larson (1999); Montenbruck and Gill (2001); Fortescue et al. (2011); Vallado (2013). Компютърните симулации позволяват срока на подготовката на космическите мисии да се съкрати, като се намали и цената (Wertz & Larson, 1999). Оптимизират се параметри на разработваните инструменти и експерименти, подобрява се качеството на решаваните научни, инженерни и технологични задачи. Нарастващият интересът към разработка на специализирани програмни системи и среди за разработка на съставни модели (multiphysics) за симулация личи от работите на Rifai et al. (1998); Heath & Jiao (2004); Tóth et al. (2005a); Tóth et al. (2005b) ; Tóth et al. (2012). Ще посочим някои примери: изчислителна аеро-термодинамика (Gnoffo, 2007), симулация на електродинамични спътникови експерименти (Li et al., 2017), динамика на частици в различни флуидни среди (Bartuschat, 2015), симулация на космически апарати и системи (Rabelo et al., 2013), климатични изследвания (Drake at al., 2005). Тенденции и бъдещи насоки в развитието на екзафлопните изчисления и разработки на средства за детайлни и всеобхватни (multi-physics, multi-scale) симулации са отразени от Shafto et al. (2012); Vázquez et al. (2016); Alowayyed et al. (2017). Heath and Jiao (2004); Tóth et al. (2005a); Tóth et al. (2005b) разглеждат настоящите и бъдещите тенденции в областта на компютърните симулациите с използване на паралелни компютърни архитектури и разработки в областта на космическите изследвания. Текущите и предстоящи посоки на технологично развитие в областта на разработки на програмни средства за проектиране на космически мисии, мулти-физични и интер-дисциплинарни модели са разгледани в публикация на National Academies of Sciences, Engineering, and Medicine (2016) и от Larson et al. (2005).

1.2. Описание на физически системи

Всяка конкретна физическа система **S** се характеризира с набор от свойства, отразяващи поведението на моделирания обект и отчитащи условията на функциониране при взаимодействие с външната среда **E** (Советов и Яковлев, 2001). Формалното представяне на функционирането на реална физическа система се основава на характеристики, попадащи в следните подмножества:

- $x_i \in X, \quad i = \overline{1, n_x}$ – входни въздействия,
- $v_l \in V, \quad l = \overline{1, n_V}$ – въздействия от външната среда,
- $h_k \in H, \quad k = \overline{1, n_H}$ – вътрешни параметри на системата,
- $y_j \in Y, \quad j = \overline{1, n_Y}$ – изходни реакции на системата.

Една реална физическа система S може да се представи като съвкупност от **N** на брой взаимодействащи или невзаимодействащи помежду си подсистеми $\boldsymbol{S_n}$ ($n =$

$\overline{1,\mathbf{N}}$). Най-общо, съвкупността от абстрактни математически модели $\mathbf{M}_n$ описващи подсистемите $\mathbf{S}_n^*$ на една реална физическа система **S** представлява агрегат (съвкупност от елементи) (Советов и Яковлев, 2001). От гледна точка на компютърното симулиране една достатъчно развита във функционално отношение програмна система представлява агрегат включващ някакъв брой алгоритми под формата на крайни автомати. Прилагането на този подход при разработката на система за симулиране позволява постигане на по-добра функционалност, ефективност, възможност за поддръжка и развитие.

1.3. Някои типове задачи, възникващи при проектиране на космически мисии

Тук ще се спрем на някои основни типове задачи, които са важни при проектиране на космически експерименти и измервания (Wertz & Larson, 1999):

- **Прогнозиране на движението** на космически обекти в околоземното пространство. На основата на аналитични или числени методи се прогнозира движението на разглежданите обекти, определят се координатите, компонентите на скоростта и ориентацията. Извършва се на различни етапи от подготовката и провеждането на космическите мисии (Wertz & Larson, 1999, pp. 47 – 69, pp. 98 – 123),
- **Траекторни изчисления**; за орбитално привързване на измервания, освен координатите и скоростта на спътниците, се изчисляват и моделни локални стойности на различни параметри на средата (атмосферни и йоносферни параметри, стойности на магнитно и електрично полета, космични лъчи), както и геометрични величини (височина над земната повърхност, координати на подспътникова точка и др.) (Wertz & Larson, 1999, pp. 95 – 130, 203 – 221).
- **Ситуационен анализ**; въз основа на различни ограничения от геометричен и физичен характер се определят времеви интервали, в които се изпълняват определени условия, важни за осъществяване на измервания и експерименти в космоса (Nazirov & Prokhorenko, 1996; Nazirov & Prokhorenko, 1998; Wertz & Larson, 1999, pp. 98 – 123; Vallado, 2013, pp. 277 – 294, 837 – 908).
- **Симулации на работата на отделни уреди, системи и динамични експерименти**; функционално моделиране (симулация на информационни потоци, температурни режими, електрически разряди, движение на зрителното поле на оптически инструменти), симулация на космическата среда (Kinnison et al., 1990; Tóth et al., 2005a), активни експерименти (със заредени частици – динамика и процеси) и др. (Wertz & Larson, 1999, pp. 241 – 291, pp. 353 – 497; Eickhoff, 2009).

Понастоящем различни модели се обединяват в рамките на общи пространствено-времеви мащаби за компютърни симулации на явления и процеси, свързани с обекти в различни области на Земята и в космоса. Известна е цяла нова дисциплина (multiphysics), която се занимава с приложението на тези модели в различни симулации и с разработка на модели и тяхното обединение за ефективно използване.

1.4. **Основни етапи в подготовката и реализацията на космически мисии и експерименти**

Могат да бъдат посочени три основни етапа за всяка космическа мисия (Eickhoff, 2011; Vallado, 2013):

a) Концептуално проучване,
b) Изследване и развитие,
c) Изпълнение.

На етап а) чрез компютърни симулации се оценяват възможностите за реализация на отделните научни задачи въз основа на предлагани идеи за провеждане на експерименти и измервания с преди използвани експериментални постановки или уреди, както и с такива, които тепърва ще се разработват. Анализират се и подбират оптималните условия за провеждане на експериментите с цел постигане на максимална достоверност на получените резултати, от гледна точка на орбитални параметри и начало на мисията. С компютърно симулиране се цели понижаване на финансовите разходи и намаляване на времето за подготовка на космическите мисии (Wertz & Larson, 1999).

Вторият етап б) е свързан с разработката и изработването на инструментите и подготовката на цялата спътникова платформа, разработка и тестване на изчислителните алгоритми. Някои автори разделят този етап на подетапи (Eickhoff, 2011).

Третият етап с) е свързан с управление и експлоатация на всички ресурси, свързани с конкретната мисия, разположени в космоса и на Земята, за решаване на научни, научно-приложни и технологични задачи. Методите на компютърно симулиране са от изключителна важност при решаване на аварийни ситуации. В последно време се оценява опасността от сблъскновение с управляеми или неуправляеми обекти (space debris) (Klinkrad, 2006; Aida et al., 2010; Назаренко, 2013).

1.5. **Модели и теории за движение на обекти в околоземното пространство**

1.5.1. **Уравнение на движение на спътник на Земята**

Основното уравнение на динамиката относно материален обект **n** с маса $\boldsymbol{m_n}$ и радиус-вектор $\vec{\boldsymbol{r}}_{\boldsymbol{n}}$ в околоземното пространство може да бъде представено в правоъгълни координати:

(1). $$\mathrm{m}_{\boldsymbol{n}} \frac{\mathrm{d}^2 \vec{r}_n}{\mathrm{dt}^2} = -\mathrm{G} \frac{\mathrm{m}_n \mathrm{M}}{\boldsymbol{r}_{\boldsymbol{n}}^3} \vec{\boldsymbol{r}}_n + \vec{\boldsymbol{f}}_n$$

Основният израз в дясната страна на векторното уравнение определя т.н. централна сила. Вторият член $\vec{f}_n$ описва смущения от различни фактори:

$$\vec{\boldsymbol{f}}_n = \vec{\boldsymbol{f}}_n^{grav} + \vec{\boldsymbol{f}}_n^{atm} + \vec{\boldsymbol{f}}_n^{Sun} + \vec{\boldsymbol{f}}_n^{Moon} + \vec{\boldsymbol{f}}_n^{light} + \vec{\boldsymbol{f}}_n^{el\,d},$$

където $\vec{\boldsymbol{f}}_{\boldsymbol{n}}^{\boldsymbol{grav}}$ – смущения в гравитационното поле на централното тяло свързани с неправилната му форма и неравномерно разпределение на плътността; $\vec{\boldsymbol{f}}_n^{atm}$ – смущение от атмосферата; $\vec{\boldsymbol{f}}_n^{Sun}$ – гравитационно привличане от Слънцето; $\vec{\boldsymbol{f}}_n^{Moon}$ – гравитационно привличане от Луната; $\vec{\boldsymbol{f}}_n^{light}$ – налягане на светлината от

Слънцето; $\vec{f}_n^{el\,d}$ – електродинамични смущения. С понижаване на реда на (1) се получава система от две векторни уравнения:

(2). $\left\| \begin{matrix} m_n \frac{d\vec{V}_n}{dt} = -G\frac{m_n M}{{r_n}^3}\vec{r}_n + \vec{f}_n \\ \frac{d\vec{r}_n}{dt} = \vec{V}_n \end{matrix} \right.$, $\vec{V}_n$ – скоростта на обекта.

1.5.2. **Смущаващи движението сили**

Бордовицына (1984, сс. 14 – 25) разглежда подробно основните смущения и модели за тяхното изчисляване. Montenbruck and Gill (2001, p. 55), илюстрират със схеми и дават подробен анализ на различните смущения в зависимост от разстоянието, моделите за тяхното изчисляване и влиянието върху прогнозирането на движението.

1.5.2.1. *Несферичен гравитационен потенциал на Земята*

Съгласно Montenbruck and Gill, 2001, гравитационния потенциал се представя във вида:

$$U = \frac{GM_\oplus}{r}\sum_{n=0}^{\infty}\sum_{m=0}^{n}\frac{R_\oplus^n}{r^n}P_{nm}(sin\varphi)(C_{nm}\cos(m\lambda) + S_{nm}\sin(m\lambda)).$$

За коефициентите C_{nm} и S_{nm} имаме:

$$C_{nm} = \frac{2-\delta_{0m}}{M_\oplus}\frac{(n-m)!}{(n+m)!}\int\frac{s^n}{R_\oplus^n}P_{nm}(sin\varphi')\cos(m\lambda')\rho(s)d^3s$$

$$S_{nm} = \frac{2-\delta_{0m}}{M_\oplus}\frac{(n-m)!}{(n+m)!}\int\frac{s^n}{R_\oplus^n}P_{nm}(sin\varphi')\sin(m\lambda')\rho(s)d^3s$$

Cunningham (1970, cited in Бордовицына и Авдюшев, 2016, сс. 93 – 98) предлага рекурентни формули за изчисляване на сферичните функции. Други рекурентни формули са получени от Дрожинер – Брумберг (Брумберг, 1980, цит. в Бордовицына и Авдюшев, 2016, сс. 98 – 101) и Холшевников (Холшевников и соавт., 2005, цит. в Бордовицына и Авдюшев, 2016, сс. 101 – 102).

В последно време са осъществени специални спътникови мисии **CHAMP** (Challanging Minisatellite Payload; начало 2000 г.), **GRACE** (Gravity Recovery And Climate Experiment; начало 2002 г.) и **ESA GOCE** (Gravity Field and steady-state Ocean Circulation Explorer; 2009 г.) за изучаване на гравитационното поле на Земята. Развитието на гравитационните модели е представено от Montenbruck and Gill, 2001, pp. 62 – 65 и Vallado, 2013, pp. 601 – 603.

Несферичността на Земята, и разпределението на плътността ѝ определят коефициентите C_{nm} и S_{nm} в горните формули. Разглеждат се късопериодични, дългопериодични и векови изменения в орбиталните параметри (Эльясберг, 1965, сс. 306-318, сс. 322 – 361; Эскобал, 1970, сс. 380 – 391; Аксенов, 1977, сс. 149 – 211; Vallado, 2013, pp. 658 – 665).

С цел увеличаване на изчислителната ефективност са разработени модели, основавани на точкови маси с подходящи размери и пространствено разпределение в рамките на обема на централното тяло (Needham, 1970; Мещеряков и Марченко, 1982; Яшникова, 2011; Russell and Arora, 2012). Съвременни разработки с използване на графични ускорители (GPU) дават

съществени преимущества на тези модели пред класическите (Russell and Arora, 2012). Разработени са също кубично-сферични модели (Jones et al., 2010).

1.5.2.2. *Аеродинамично съпротивление от атмосферата*

За съпротивлението от атмосферата имаме (Vallado, 2013, pp. 551-555):

$$\vec{f}_{drag} = -.5\rho_{atm} c_{drag} A_{mid} v_{rel}^2 \frac{\vec{v}_{rel}}{|\vec{v}_{rel}|},$$

където $\boldsymbol{\rho_{atm}}$ – плътност на атмосферата определена чрез модел, c_{drag} – безразмерен коефициент на съпротивление, $\boldsymbol{A_{mid}}$ – средно („миделево") сечение на спътника, перпендикулярно на вектора на скоростта и $\vec{\boldsymbol{v}}_{\boldsymbol{rel}}$ – скорост на обекта относително атмосферата.

Най-проблематично е определянето на атмосферната плътност въз основа на атмосферни модели (Montenbruck and Gill, 2001; Vallado, 2013; Vallado and Finkleman, 2014; Бордовицына и Авдюшев, 2016). Vallado (2013) и Vallado and Finkleman (2014) разглеждат регулярни и нерегулярни вариации в плътността на земната атмосфера, като се обръща внимание на правилното използване на динамичните модели на атмосферата.

1.5.2.3. *Смущения от трето тяло*

При движение на обекти около Земята като „трето тяло" се разглеждат основно Луната и Слънцето.

За смущение от Луната и Слънцето съответно имаме:

$$\vec{f}_{Moon} = -GM_{Moon}\left(\frac{\vec{r}_{sat} - \vec{r}_{Moon}}{|\vec{r}_{sat} - \vec{r}_{Moon}|^3} - \frac{1}{|\vec{r}_{Moon}|^3}\right)$$

$$\vec{f}_{Sun} = -GM_{Sun}\left(\frac{\vec{r}_{sat} - \vec{r}_{Sun}}{|\vec{r}_{sat} - \vec{r}_{Sun}|^3} - \frac{\vec{r}_{Sun}}{|\vec{r}_{Sun}|^3}\right)$$

В тези изрази $\vec{\boldsymbol{r}}_{\boldsymbol{Moon}}$ или $\vec{\boldsymbol{r}}_{\boldsymbol{Sun}}$ са радиус-вектори на смущаващи обекти. В изчислителната практика, в случая на Луната, се използват опростени формули съгласно теорията на Hill-Brown (Прохоренко, 1976; Vallado, pp. 287 – 289). За определяне на положението на Слънцето също се използват опростени формули (Прохоренко, 1976; Vallado, pp. 277 – 280).

Значението на тези смущения за подобряване на точността на прогнозиране на орбитите е при височини над 1000 км. Montenbruck and Gill (2001, p. 55) илюстрират порядъците на различните видове смущения в зависимост от височината на спътниците над земната повърхност.

1.5.2.4. *Смущаващи сили от светлинно налягане*

Montenbruck and Gill (2001) и de Iaco Veris (2018) представят физическа обосновка на това смущение. Това смущение се представя така:

$$\vec{\boldsymbol{f}}_{srp} = -\rho_{sr} c_r A_{Sun} \frac{\vec{\boldsymbol{r}}_{sat-Sun}}{|\vec{\boldsymbol{r}}_{sat-Sun}|},$$

където ρ_{sr} – потокът падаща слънчева радиация [W/m^2], c_r – коефициент на отражение на повърхността на спътника; зависи от материала и ориентацията на спътника, A_{Sun} – сечението на спътника, перпендикулярно на посоката към Слънцето, $\vec{\boldsymbol{r}}_{\boldsymbol{sat-Sun}}$ – вектор определящ посоката към Слънцето.

При спътници с големи панели, е необходимо отчитане на ориентацията (Иванов и соавт. 2016, с. 87). Промените в потока слънчева радиация при слънчеви максимуми предизвикват смущения в движението на спътниците по-големи от атмосферното съпротивление (Vallado, 2013, p. 555).

При преминаване през сянката на Земята смущението отсъства. Ferraz-Mello (1963, 1964, cited in Бордовицына, 1992) описва сянката на Земята с цилиндричен модел за подобряване на орбитите на спътници. Бордовицына и соавт. (1992) разглеждат коничен модел с полусянка.

1.5.2.5. *Други смущения*

Смущения с малки амплитуди (в сравнение с разгледаните по-горе) се предизвикват от:

- Прецесия и нутация на екваториалната равнина (Аксенов, 1977, сс. 311 – 319), движение на полюсите (de Iaco Veris, 2018, pp. 356 – 361); необходимост от преизчисляване на гравитационните модели;
- Приливна деформация на мантията, океански и атмосферни приливи, причинени от Луната (Аксенов, 1977, 320 – 327; Vallado, 2013, pp. 585 – 587);
- Привличане от атмосферата (Аксенов, 1977, 327 – 328);
- Релативистки поправки (Аксенов, 1977, 331 – 332; de Iaco Veris, 2018, pp. 543 – 548).

1.5.3. **Числени методи за интегриране на системи от уравнения в астродинамиката**

В астродинамиката съществуват два основни типа задачи, свързани с интегриране на уравнения на движения – с **начални стойности** и **гранични задачи** (Абалакин и соавт., 2012). При задачи с начални стойности се търси неизвестната функция $\vec{r}(t)$, при известни $\vec{r}(t_0)$, както и първата ѝ производна $\dot{\vec{r}}(t_0)$ в начален момент $t_0 (t_0 < t)$:

$$\text{I.} \quad \begin{cases} \ddot{\vec{r}} = \boldsymbol{f}(\vec{r}, \dot{\vec{r}}, t) \\ \vec{r}_0 = \vec{r}(t_0);\ \dot{\vec{r}}_0 = \dot{\vec{r}}(t_0) \end{cases}$$

Уравнение (I) няма аналитично решение поради сложността и спецификите в смущаващите функции. Съществуват различни методи за числено решаване на системи от диференциални уравнения прилагани в астродинамиката (Montenbruck & Gill, 2001; Авдюшев, 2010). Montenbruck and Gill (2001, p. 117) посочват, че всеки отделен метод притежава специфични предимства и недостатъци, но няма най-добър, подходящ във всички случай.

1.5.3.1. Едностъпкови методи за интегриране

1.5.3.1.1. *Методи за решаване на **ОДУ** от първи ред*

а). *класически методи на Рунге Кута*

Класическия метод на РК е от $4^{\text{-ти}}$ порядък. Shanks (1965, 1966) увеличава точността на метода с разработка на схема от $8^{\text{-ми}}$, а Hairer (1978) разработва метод от $10^{\text{-ти}}$ ред. Curtis (1975) разработва схема с 18, докато методът на Hairer, 1978 е със 17 пресмятания на функциите в рамките на една стъпка.

б). *методи на Рунге Кута Фелберг с оценка на грешката*

За всяка стъпка от интеграционния процес се пресмятат две решения чрез методи с различна точност, единия от ред $\mathbf{p}$ и другия от ред $(\boldsymbol{p}+1)$:

$$y_i = y_i^0 + h.\sum_{k=0}^{n} c_k g_{i,k} + O(h^{p+1}),\ \bar{y}_i = y_i^0 + h.\sum_{k=0}^{n+1} \bar{c}_k g_{i,k} + O(h^{p+2})$$

Методите от редове p и $(p+1)$ се разработват с различни коефициенти c_k и $\bar{c}_k$, но междинните точки в които се изчисляват функциите $g_{i,k}$ са едни и същи. По този начин, решението от ред $(p+1)$ се получава с минимални допълнителни изчисления.Така разликата между двете решения дава оценка на главния член на грешката:

$$\bar{O}_i \approx \sum_{n=0}^{n+1} (c_k - \bar{c}_k).\, g_{ik}$$

Развитието на методи от типа Runge-Kuta-Fehlberg е в посока към повишаване на точността на схемите за интегриране, чрез оптимизиране на коефициентите (Verner, 1978; Prince and Dormand, 1981; Tsitouras and Papakostas, 1999), повишаване на реда на методите и намаляване на междинните точки в които се изчисляват функциите.

Специално ще посочим използване на схеми от различен ред при интегриране на системи от диференциални уравнения. Shampine et al. (1978) използва две схеми от ред 4(3) и 8(7) и установява предимствата на интегрирането с оптимална схема. Друга възможност е разгледана от Cash & Karp (1990), които използват повече схеми с различен ред.

в). *метод на Еверхарт*

Методът на Everhart (Everhart, 1974) се основава на друг подход (Gauss-Radau), в сравнение с Runge-Kutta, при избора на вътрешните точки в които се изчисляват функциите. Разработени са варианти от 15-ти и 27-ми порядък (Авдюшев, 2010). Анализ и оценка на възможностите на метода, както и сравнение с други методи, са извършени от Бордовицына (1984); Бордовицына и соавт. (1992).

1.5.3.1.2. *Методи за решаване на **ОДУ** от втори ред с оценка на грешката*

Нистрьом (Evert Johannes Nyström, 1895 – 1960) разработва методи за решаване на **СОДУ** от втори (и по-висок) ред, които не изискват предварително понижаване на реда на уравнение (1) от §1.5.1 (Nyström, E.J., 1925). Те обаче изискват дясната страна на системите да не зависят явно от първата производна на търсената функция.

Методи на Нистрьом с различен порядък на точност са предложени от различни автори (Fehlberg, 1972; Filippi & Gräf, 1986; Dormand & Prince, 1987). Ефективността на методи на Nyström при решаване на астродинамични задачи е оценена от Бордовицына (1981) и Montenbruck and Gill (2001, p. 130 – 132).

1.5.3.2. Многостъпкови методи

Разработени са класове от многостъпкови методи Adams-Bashforth (явни) и Adams-Multon (неявни) (Vallado, 2013) с използване на стойности на неизвестните функции за повече минал моменти от времето. За всяка следваща стъпка по

времето се използва по една оценка на неизвестните функции. Различни автори (Бордовицына, 1984; Montenbruck and Gill, 2001; Авдюшев, 2010; Vallado, 2013; de Iaco Veris, 2018) изчерпателно излагат различни аспекти (теория, сравнителни оценки на ефективност, силни и слаби страни) свързани с многостъпкови методи за интегриране на **СОДУ**.

1.5.4. Методи за интерполация, интерполанти

Твърде често, вместо стойностите на векторите $\vec{\boldsymbol{r}}(\boldsymbol{t})$ и $\vec{\boldsymbol{v}}(\boldsymbol{t})$ в моменти, определени от самото числено решение $\boldsymbol{t = t_0 + \Delta t}$, са необходими съответните стойности в междинни моменти $\boldsymbol{t_k} \in [\boldsymbol{t_0}, \boldsymbol{t_0 + \Delta t}]$. Това се налага, когато се търсят определени орбитални събития или за графично представяне на движението на спътниците (de Iaco Veris, 2018, p. 924).

Horn (1983) прави решителна стъпка напред, като развива методи на Рунге-Кута от трети, четвърти и пети ред, като с минимални допълнителни изчисления, наред с резултатите за неизвестната функция в края на стъпката (интервала на интегриране), определя и стойности в L междинни точки за $\boldsymbol{t = t_0 + \sigma.\Delta t}$, $\sigma\epsilon[0,1]$, $\sigma = \frac{1}{l}$.

Papageorgiou and Tsitouras (1989); Hairer, Nørsett and Wanner (1993, pp. 188-195), Montenbruck and Gill (2001, pp. 127 – 129), Tsitouras (2007) и de Iaco Veris (2018, pp. 924-933) представят подробен преглед на разработваните методи на Runge-Kutta-Fehlberg/Nystrom за интегриране с интерполиране в междинни точки.

1.5.5. Връзка между точността на методите за интегриране и прилаганите модели за смущения

Използването на числен метод от по-висок ред позволява да се отчитат по-слаби смущения (Бордовицына и Авдюшев, 2016, с. 115 – 117). По-точните модели на гравитационното поле, включващи хармоници от по-висок ред, отчитат по-дребномащабни пространствени осцилации, които пък за да бъдат отчетени от метода за интегриране изискват подходяща, достатъчно малка стъпка във времето (Vallado, 2013, p. 536)!

1.6. Ситуационен анализ на космически експерименти и мисии

1.6.1. Аналитични методи за ситуационен анализ

Различни автори се занимават с изследването и решаването на т.н. „уравнение на сянката“ (Еременко, 1965; Соколов, 1980; Ortiz Longo & Rickman, 1995; Neta and Vallado, 1998). Друг проблем с аналитично решение е свързан с определяне на момента на минимално сближаване между два обекта на околоземни орбити (Hoots, Crawford and Roehricg, 1984). По-нататък, в решаване на този проблем, имащ отношение към стълкновения между различни обекти в околоземното пространство, имат принос и други автори (Kholshevnikov and Vassiliev, 1999).

Atanasov (1992) предлага общ аналитичен метод за ситуационен анализ, основан на трансформация на ситуационното условие към кеплеровата равнина. Методът е приложен също за оценка на преминаване на спътник над кръгов сектор от земната повърхност (Atanassov, 2003). Atanassov (2009) прилага метода и за определяне на преминаването през ударната вълна и магнитопаузата.

Аналитичните методи са бързи, но с ограничени възможности за прилагане, поради ниската точност на кеплеровото решение. Освен това, те изискват квазистационарност за геометричните модели на ситуационните условия в рамките на времето за една обиколка на спътника. След анализа за всяка обиколка се прави корекция на орбиталните елементи и поправки за геометричния модел на ситуационното условие.

1.6.2. Ситуационен анализ основан на дискретизация по времето

На основата на дискретизация чрез числени методи на решението на уравнение (1) от §1.5.1 могат да бъдат решавани ситуационни задачи с произволна сложност, висока точност и динамични ситуационни условия. За пример ще посочим мисията **ИНТЕРБОЛ**, включваща два основни и два субспътника (Прохоренко, В.И., 1985). Анализът е свързан с установяване на времената за измерване при преминаване на спътниците през различните области на магнитосферата.

Гайдаров и соавт. (1989) използват дискретизация на кеплерово приближение за получаване на вектора на състоянието на станция Мир за ситуационен анализ на астрофизически експерименти, на основата на 5 ситуационни условия, свързани със Слънцето, Луната, радиационния фон и ориентацията.

1.7. Планиране на спътникови операции

1.7.1. Цели на планирането

- постигане на максимална ефективност и натоварване на бордовите и наземни системи с оглед на постигане на максимална възвращаемост за доставчика на услугите;
- постигане ма максимално натоварване от гледна точка на потребностите на максимален брой потребители на услугите.

1.7.2. Понятие за планиране и изготвяне на разписание

Проблемът за изготвяне на разписание за изпълнение на различни активности, стоящи в основата на изпълнение на научни и системни задачи, се основава на следните четири основни типа обекти (Pemberton & Galiber, 2001):

- **Задачи** – действия, които трябва да бъдат изпълнени,
- **Ресурси –** хора, спътници, инструменти,комуникационни средства (и всички останали средства), които са необходими за изпълнение на задачите,
- **Събития** (орбитални събития) **–** определят областта в пространството или времето, когато задачата може да бъде изпълнена,
- **Ограничения –** това са допълнителни ограничения върху изпълнимостта на задачите, свързани с количеството ресурси или взаимодействие с други задачи.

Ограниченията могат да бъдат три типа (Pemberton & Galiber, 2001):

- **Ограничения на задачите** – те са свързани със зависимости между задачите. Задача, свързана с регистрация, трябва да е предшествана от задача за насочване. Задача за трансфер на данни към наземна станция може да се осъществи след задача по събиране на данните.
- **Ресурсни ограничения.** Например, не е възможно да се извършат едновременно две задачи свързани с регистрация, ако уредът е един. Някои задачи могат да изискват повече от един ресурс – например, инструмент за

измерване и памет за временно съхранение на информацията, комуникационни средства. Ресурсите могат да бъдат възстановими или невъзстановими.

– **Ограничения на събития –** те определят **времевите интервали** (времеви прозорци), когато една задача може да бъде изпълнена.

1.7.3. Планиране на свръхзаявени спътникови операции

Ситуация, в която подадените заявки, пораждащи изпълнение на спътникови операции, са свръх възможностите за изпълнение, се получава при наличие на ограничени ресурси, голям брой цели за наблюдение, повече спътници за обслужване от наземни станции. В литературата са разгледани различни аспекти и подходи за планиране на операции при свръхзаявки (Smith and Pathak, 1992; Miller et al., 1988; Barbulescu et al., 2004; Kramer and Smith, 2006).

1.7.4. Планиране на астрофизически експерименти

За планиране на операции, свързани с астрофизически наблюдения, се определят интервалите от време, в които е подходящо наблюдение на заявени астрофизически обекти. Характерно е изискването за изпълнение на редица ограничения (Johnston, 1992). Те са свързани с използване на оптически инструменти, откъдето следват ъглови ограничения относно направлението на зрителната ос към желания за наблюдение обект и ярки източници, като Слънцето и Луната. За целите на ориентацията на платформата с научни инструменти се използват ограничения спрямо водещи звезди. С цел екраниране на слънчева светлина и поддържане на топлинни режими (когато платформата обикаля около Земята) може да се наложи изискване за извършване на наблюдението в сянката на Земята, при допълнителни изисквания за намиране на Слънцето и Луната на определени ъгли под хоризонта. Важно е също и изискване към радиационния фон (да е под определена стойност), с цел защита на електрониката. Оптималното планиране на последователно изпълнение на задачи, свързани с наблюдение на различни области от небесната сфера, може да изисква ограничаване на ъгъла на пренасочване между двете направления. Времето за пренасочване (зависи от разположението на обектите за наблюдение върху небесната сфера) се отчита в процеса на планиране. Планирането на операции при дистанционни изследвания на Земята може да има общо с казаното по отношение на астрофизическите изследвания.

1.7.5. Методи за планиране на спътникови операции

Това са комбинаторни методи за достигането на оптимално (или близко до него) решение на задачата за планиране на спътникови операции, с оглед на специфики, произтичащи от различни ограничения. „Лакоми“ алгоритми за планиране на операции са лесни за реализация. Въпреки че решението може да не е абсолютно оптимално, то отговаря на локален максимум. При тези алгоритми всяка следваща операция се проверява, като се сравнява с вече планирана. Сравненията се основават на различни евристични правила, например:

– възходящ ред на началните моменти за всяка операция,

– възходящ ред на крайните моменти,

– възходящ ред на времетраенето на операциите,

– низходящ ред на конфликтите (застъпване по време) на поредната операция с оставащите.

Прилагане на „**greedy**“ алгоритми за планиране на спътникови операции е разгледано от Pemberton (2000) и Wolfe & Sorensen (2000).

1.8. **Нерегулярни изчисления**

Нерегулярни алгоритми се разглеждат в различни научни области (Raghavachari & Rogers 1996; Gutiérrez, et al., 2000). Това са изчисления с разредени матрици, динамика на флуиди, обработка на изображения, молекулярна динамика, симулации на галактики и купове от звезди и галактики, астрофизика, моделиране на климата, оптимизационни задачи, вихри и гранични токове в океански течения, симулация на потоци в разредени течности, разпространение на електромагнитни вълни през среди, молекулярна динамика в течна среда, локални деформации в геологически системи.

Във всички случаи нерегулярността се изразява в някакъв вид неоднородност на математическите или програмни модели за пресмятания или представяне на данните. Нерегулярностите могат да намалят ефективността от прилагане на паралелни изчисления, поради неравномерното натоварване на процесорите (load balancing).

1.9. **Хардуерни и софтуерни средства за паралелни изчисления**

1.9.1. **Хардуерни архитектури**

1.9.2. **Паралелни процесорни архитектурни свойства**

а). Паралелизъм на ниво битове.
б). Паралелизъм на ниво процесорни инструкции (pipelining, instruction level parallelism); развива се от средата на 1980$^{-те}$.
в). Супер-скаларност – (instruction-level parallelism – ILP);
г). Паралелизъм по данни.
д). Промяна на реда на изпълнение на инструкциите (Out-of-order execution)**.**
е). Кеш памет с по-голям размер.
ж). Многонижковост (multithreading & hyper threading).
з). Многоядрени процесори (multi-core, many-core).
и). Ускорители (копроцесори Xeon pHi, графични процесори GPU).

1.9.3. **Паралелни компютърни архитектури** (според организацията на паметта)

– **Системи с обща памет** (shared memory- споделено адресно пространство).
– **Системи с разпределена памет** (distributed memory system). Множество компютри (възли) обединени от локална мрежа. Всеки от тях представлява система с обща памет.
– **Системи с глобално адресно пространство** (global address space). При условията на системи с разпределена памет се разработват операционни системи и езици които третират отделните оперативни памети като една обща.
– **Суперкомпютри**. Това са компютърни системи с голям брой процесори, които са свързани помежду си с бързи мрежи. За пример, българският суперкомпютър Авитохол се състои от 121 сървърни единици с по два 8-ядрени процесора Xeon и един Xeon pHi.

1.9.4. **Програмни модели за паралелизация на алгоритми**

1.9.4.1. статични модели

- **конвейеризация** (pipelining). При този модел независими части от изпълнимия код се преобразуват в изчислителни нишки, като управлението към всяка от тях се предава в определен момент, който зависи от достигане до точка в родителския или някой от дъщерните нишки,
- **разклонение** – **свързване** (fork – joint); рекурсивно прилагане на модела на няколко нива на вложеност. Тук също нишките са в отношение главна-подчинени (master – workers/slaves). Стартирането на дъщерните нишки става в определена точка от родителската; в тази точка се чака всички нишки да завършат работата си,
- **вложен паралелизъм** (цикли). Прилага се при цикли, когато отделните итерации са независими помежду си.

Статичните модели са приложими когато предварително е известен размерът на подзадачите, които ще се изпълняват от подчинените нишки или броят им е малък. Възможно е отделните нишки да завършват изчисленията си по различно време при което някои от процесорите да не са напълно използвани.

1.9.4.2. Динамични модели

Прилагането на програмен модел „пул от нишки" (pool of threads) е пример за динамично планиране на изчисленията. При тази организация всяка нишка взема част от цялата изчислителната задача и се обръща за нова след като приключи с предишната. Основно изискване е две нишки да не обработват една и съща подзадача! Необходима е синхронизация между нишките при вземане на поредната подзадача. Този модел е подходящ при нерегулярни изчисления, когато цялата задача се разделя на подзадачи с различен, отнапред неизвестен размер. При този модел броят на нишките е обикновено много по-малък от броя на подзадачите. Важно е да се помни, че броят на нишките не е желателно да надхвърля броя на достъпните процесори в системата.

1.9.5. **Метрики за определяне на ефективността от паралелизация**

Широко прилагани са популярните метрики за оценка на ускоряването S_n и ефективността E_p от прилагане на паралелизация (Xian-He and Gustafson, 1991):

(1). $S_p = \frac{T_s}{T_p}$

(2). $E_p = \frac{T_s}{(p.T_p)}$

В последните формули T_s е времето на изпълнение на оптималния сериен код, T_p е времето за изпълнение на паралелния код при използване на p процесора.

1.10. Програми за симулация на космически мисии

Разгледани са различни известни програми за анализ на космически мисии в околоземното пространство – **STK, FreeFlyer, Gmat**. Има също програми за проектиране на космически мисии свързани с Луната и други планети от слънчевата система (**SPICE**). Споменати са програми за анализ на опасността от

електрични разряди (**NASCAP, ECO-M, DICTAT, MUSCAT**) и термични проблеми (**MSC** – **Nastram**).

1.11. Изводи към обзора

От извършения обзор следват изводи по отношение на методите за интегриране, ситуационния анализ и паралелните алгоритми и изчисления:

- Методите за числено интегриране на **RKF** са универсални, лесни за програмиране, намират широко приложение и имат версии с интерполанти;
- Ситуационният анализ е интердисциплинарна област, която подлежи на формализация и разработка на формални и оптимизирани методи;
- Разработката на сложни модели за компютърни симулации е в тясна връзка с прилагане на паралелни изчисления.

Глава II. Разработка на методи и изчислителни инструменти

2.1. Интегратор на системи от обикновени диференциални уравнения

2.1.1. Избор на метод от оптимален ред

Под избор на метод от оптимален ред ще разбираме такъв, който удовлетворява изискванията за точност на решението при дадена постоянна стъпка на интегриране по времето при минимални изчислителни разходи и осигуряване на устойчивост на решението в рамките на интервала на интегриране. Изборът се прави в рамките на класическата фамилия от методи на Fehlberg с ред на точност 1/2, 2/3, 3/4, 4/5, 5/6 и 7/8.

2.1.1.1. Стратегия за избор на метод с оптимален ред

Въпросът за избор на схема с оптимален ред на точност беше разгледан относно решаване на ОДУ с дясна страна, съдържаща екстремуми (Атанасов, 1987):

$$y'(x) = \exp(\sin x + 2 * \sin x * \cos x)$$

Интегриране с променлива стъпка е неудобно, ако е необходимо да знаем стойностите на неизвестната функция в равноотстоящи стойности на независимата променлива.

Стратегията за избор на схема с оптимален порядък на точност може да бъде представена така:

а'). $\overline{0} > \varepsilon.\Delta r$, където $\overline{0} = \sqrt{\overline{0}_x^2 + \overline{0}_y^2 + \overline{0}_z^2}$, $\Delta r = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$, а ε е стойността на относителната грешка, подбрана да осигури глобална устойчивост на решението за интервала на интегриране. Тогава се преминава към схема на интегриране с по-висок порядък на точност; когато се достигне подходяща схема, стойността на произведението $c_p.\overline{O}$ се запазва за бъдещо използване. Коефициентите c_p са емпирично подбрани, като следните стойности се приемат като подходящи: $c_2 = 0.05$, $c_4 = 0.09$, $c_5 = 0.1$, $c_7 = 0.2$.

б' $\overline{0} < \varepsilon.\Delta r$, $c_p.\overline{O}$, тогава се преминава към схема от по-нисък порядък.

Когато наличната схема с максимален ред на точност не е достатъчна за да се осигури интегриране с необходимата локална грешка, тогава се преминава към интегриране с променлива стъпка в рамките на основната. Този подход беше изпробван за интегриране на движението на спътници (Атанасов, 1986).

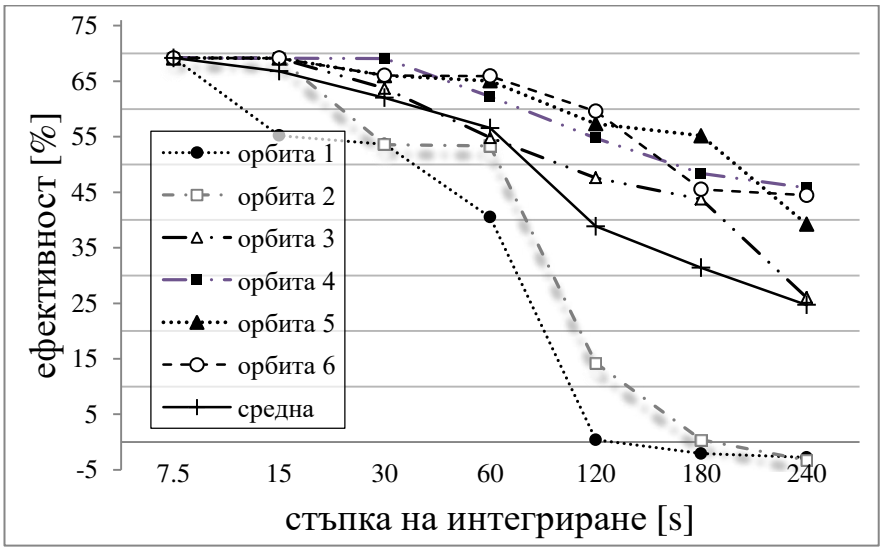
2.1.1.2. Оценка на ефективността на избор на метод от оптимален ред

За оценка на ефективността (Atanassov, 2007) са използвани 6 орбити с различни големи полуоси и ексцентрицитети, при различни стъпки на интегриране от 30 до 120 секунди.

Ефективността ***E*** е оценена по формулата:

$$E = \frac{n_{7/8} - n_{\sim}}{n_8},$$

където $n_{7/8}$ са броя на обръщенията към дясната страна на **СОДУ** при интегриране със схема с максимален порядък на точност, а $n_{\sim}$ е съответно броя при избор на оптимална схема. На фигура 2.1.1[4.1.1][1] е показана ефективността от прилагане на избора на метод с оптимален ред на точност. Освен ефективността за всяка орбита, показана е и средната ефективност.



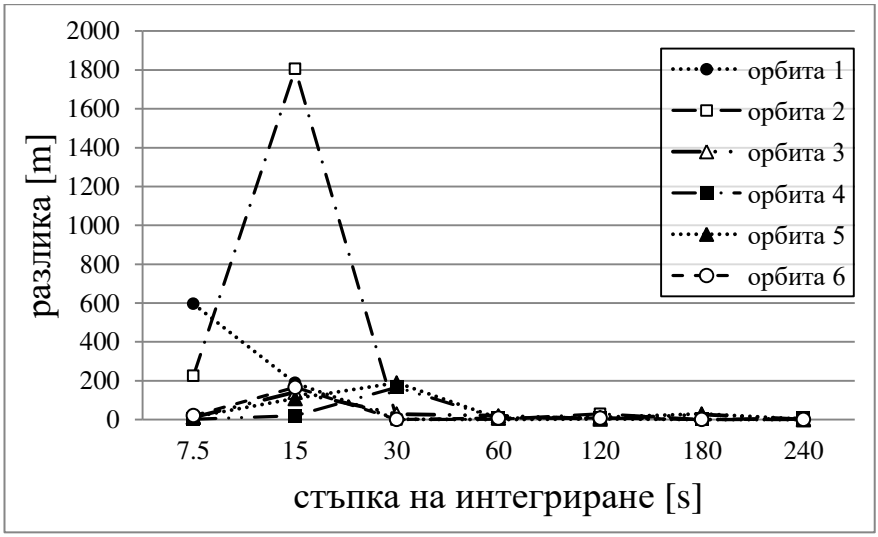
Фигура 2.1.1.[4.1.1] Ефективността от избор на оптимална схема за интегриране при различни стъпки.

Схеми от по-нисък ред удовлетворяват изискването за локална точност при малки стъпки, поради което ефективността е висока. Уравненията на движение при по-ниските орбити се интегрират със схеми от по-висок ред, поради което ефективността от автоматичния избор е по-ниска от тази при по-високите орбити. Това следва от по-голямата кривина и по-голямата площна скорост.

Ефективността е отрицателна за орбити 1 и 2 (a_1= 6 800 000 m, e_1= .01 и a_2= 7 500 000 m, e_2= .10) при стъпки по-големи от 180 сек. Това е така, понеже за тези орбити за осигуряване на зададената локална точност се налага интегриране с по-малка стъпка, дори при използване на схема от 7/8 ред. Тогава броят на пресмятанията на дясната страна на системите се удвоява.

[1] За краткост, в автореферата са показани част от фигурите, като в квадратни скоби е показан номера на всяка, в основния текст на дисертацията.

На фигура 2.1.2[4.1.2] е показана устойчивостта на решението при интегриране на същите орбити в интервал от едно денонощие. Тя е оценена чрез разликата с решение, получено при използване на схема от $7/8^{-ми}$ ред. При най-ниската орбита и стъпка 15 сек отклонението е 1800 м, което се дължи на използването на схема от нисък ред. Схемите от по-нисък ред интегрират с изискваната локална точност, но глобалната неустойчивост на решението е по-ниска.



Фигура 2.1.2.[4.1.2] Разликите между решенията получени с избор на оптимална схема и схема от $7/8^{-ми}$ ред, за време едно денонощие.

2.1.2. **Сериен вариант на интегратора**

Неформалната част на **ИСОДУ** включва подпрограми, реализиращи схемите на **RKF,** подпрограма за оценка на грешката и управляваща подпрограма, в която се извиква подпрограмата на оптималния метод. Това са подпрограми, представляващи серийния вариант на интегратора.

2.1.3. **Паралелизация на интегратора**

Можем да посочим няколко причини, поради които решаването на задачи с начални стойности, изисква много изчислително време:

- Използват се модели на движение с повече и сложни за изчисляване смущения;
- Налага се едновременно интегриране на орбитите на много обекти;
- Времевият интервал, в който се извършва интегрирането, е дълъг;
- Симулациите се повтарят многократно.

За преодоляване на посочените причини в поносими времеви интервали, освен ефективни методи за интегриране, прилагат се и паралелни изчисления.

Два подхода за паралелизация на изчисленията при числено интегриране на системи от диференциални уравнения са посочени от Gear, 1987:

- Паралелизъм през метода (Parallelism across the method);

– Паралелизъм през системата (Parallelism across the system).

Изразът „паралелизъм през метода“ изразява възможността отделни етапи в рамките на един метод да са независими и да могат да бъдат изчислявани едновременно на различни процесори. Явните методи на **RK** се основават на изчисляване на функциите $f(t_l, \vec{r}_l, \dot{\vec{r}}_l)$ (виж §1.5.3.1.1.6) за последователни моменти от време в рамките на стъпка на интегриране $t_m < t_l < t_{m+1}$. Изчисляването на $g_{i,0}$ и $g_{i,l}$ за моментите t_l се основава на изчисления за предишни моменти $f(t_{l-1}, \vec{r}_{l-1}, \dot{\vec{r}}_{l-1})$ и $g_{i,l-1}$. По тази причина тези методи не предполагат възможности за този вид паралелизъм.

Изразът „паралелизъм през системата“ се отнася за ситуация, когато един метод може да се прилага независимо към математическите модели, описващи части на сложна система, и съответните изчисления могат да се извършват на различни процесори. Изчисленията на коефициентите $g_{i,l;n}$, за **СОДУ** за всеки n$^{\text{-ти}}$ обект са независими, и могат да бъдат извършвани (едновременно) на различни процесори.

2.1.3.1. Паралелизация основана на изчислителни нишки

Ефективността от паралелните изчисления зависи от разделянето на цялата задача на подзадачи. В случая на решаване на задачи с начални стойности, включващи голям брой обекти, численото интегриране на уравненията на движение води до **нерегулярни изчисления** (виж §1.8). Причините са следните:

– Оптималният ред на метода за интегриране зависи от локалната кривина на орбитата и скоростта на движение на обекта. Близо до перигея (при елиптични орбити) или при ниски орбити е необходима схема от висок ред и обратно, около апогея и при висока орбита – по-нисък ред може да се окаже достатъчен;
– В зависимост от типа на орбитата, при интегрирането ѝ може да се приложи подходящ модел на смущенията, различен от моделите за други орбити.

Разглежданите нерегулярности (резултат от модели и методи) са променливи във времето и пространството. В този случай, на отделните подзадачи съответстват променливо в симулационното време, различни по количество изчисления (load imbalance). Прилагането на паралелизъм, основан на нишки и динамично планиране на изчисленията, основано на модела „**пул от нишки**”, решават ефективно проблема произтичащ от нерегулярността на изчисленията (Korch, M. and Rauber, T., 2004; Rauber and Rünger, 2010).

2.1.3.2. Създаване на актуален интегратор

Създаването на актуален паралелен интегратор (**АПИ**) е насочено към решаване на конкретна задача и включва:

– Създаване на определен брой нишки обединени в пул;
– Подготовка и предаване към нишките на интегратора на данни за решаваната задача и управляващи параметри за пула.

В рамките на един симулационен процес могат да бъдат създадени и да функционират няколко актуални интегратора за решаване на различни задачи. Те могат да функционират последователно, в рамките на една и съща стъпка от времето, или паралелно въз основа на специално разработен програмен модел („обединение на пулове от нишки“), който ще бъдат разгледан в §2.3.

Един клас задачи са свързани с интегриране на уравненията на движение на спътници, участващи в дадена мисия. Друг клас задачи може да бъдат свързани с определяне на движението на оперативни спътници, движещи се по орбити, с които са възможни опасни сближавания. Трети клас задачи са свързани с определяне на движенията на орбитални отломки. Различните класове от задачи могат да се отличават по смущаващите сили, които се отчитат, по използваните атмосферни модели за всяка от тях, или по подпрограмите, с които се описват десните страни на **СОДУ**. За електродинамична задача, свързана с определяне на движението на заредени частици, може да се създаде друг актуален интегратор.

Създаване на нишките за интегратора и събития за синхронизация (между нишките в рамките на пула и с родителската нишка) се извършва със специална подпрограма[2] **CreatePoolThreads** (фиг. 2.1.3[4.1.5]). Броят на нишките се задава с формалния аргумент **num_AI_threads**. Числовите идентификатори (handler) за тези събития се записват в масива **AI_thread_par**.

Създаване на „пул от нишки“ за актуален интегратор може да стане на подходящо място в родителска нишка или в диалогова подсистема на система за симулации. След създаването си, нишките на интегратора остават в суспендирано състояние до извикване на помощните подпрограми **Data_AI** и **Preparation_AI,** с които се подготвят за предаване към **АИ** на необходимите данни.

Външното име **SatelliteIntegrator** (което се съдържа в IntegratorName) е име на буферна подпрограма. Това име се предава на системната (API) функция **CreateThread**, която създава изчислителна нишка. **SatelliteIntegrator** е първата, подпрограма от изчислителната нишка, която се обръща към подпрограма, управляваща синхронната работа на нишките в интегратора (виж §2.1.3.4).

```
SUBROUTINE  CreatePoolThreads(IntegratorName, num_threads,thread_par,ha_1)
  EXTERNAL                          IntegratorName
                  …
    ha_1 = CreateEvent(security1,.false.,.true.,0) !За синхронизация между нишките
  DO    i=1,num_threads
        thread_par(2,i) = i
        thread_par(1,i) = CreateThread(security,stack,IntegratorName, &
                          LOC(thread_par(2,i)),CREATE_SUSPENDED,thread_id)
        thread_par(3,i) = CreateEvent (security1,.false.,.true.,0) ! събитие за начало
        thread_par(4,i) = CreateEvent (security1,.true.,.false.,0) ! събитие за край
  END DO;
END SUBROUTINE
```

Фигура 2.1.3.[4.1.4] Фрагмент от подпрограмата **CreatePoolThreads**. Създаване на нишките за актуален интегратор

2.1.3.3. Инициализация на актуален интегратор

За да може да се осъществи максимална функционалност на интегратора и да може да се осъществява ефективен обмен на данни между родителска нишка и нишките на интегратора се използва буферна програма (Atanassov, 2013; Atanassov, 2014b). Нейното име се предава на програмата **CreatePoolThreads**, която създава пула от нишки (фиг. 2.1.3), а оттам и на самите нишки при

[2] Всички разгледани по-нататък подпрограми са разработени на програмния език fortan 95.

създаването им със системната функция **CreateThread**. Тази буферна подпрограма получава чрез глобални данни (общи области) стойности на величини необходими за работата на нишките, адреси на масиви чрез които нишките получават началните условия и адреси на които да записват получените резултати. Освен това, в буферната програма се задава (с оператор **external**) името на подпрограмата която изчислява дясната страна на системите от диференциални уравнения. По този начин могат да бъдат решавани различни класове от задачи с различни десни страни на системите от уравнения.

Инициализацията на актуалния интегратор се извършва чрез помощни подпрограми, извиквани от родителската нишка (виж §2.1.3.5), които предават данни (начални условия, управляващи параметри и адреси) към буферна подпрограма чрез глобални данни.

След приемане на данните от буферната подпрограма (копирането им в локални променливи може да стане след поставяне на нишките в несуспендирано състояние), следва обръщение към подпрограма **Integrator**, която служи за управление на пула от нишки за конкретния актуален интегратор.

2.1.3.4. Подпрограма за управление на нишките на интегратора

За да функционират създадените нишки като пул е необходима подпрограма, която да управлява съвместната им работа, в координация с родителската нишка. По-нататък ще бъде разгледан универсален вариант на такава подпрограма, основан на полиморфизъм (виж §2.2.4.4.3, фиг. 2.2.5).

След като родителската нишка осигури запис на всички необходими за интегратора данни в съответни общи области, променя състоянието на нишките в несуспендирано. Тогава буферната подпрограма копира глобалните променливи от общите области в локални променливи. След това буферната подпрограма **SatelliteIntegrator** се обръща към подпрограмата **Integrator** и му предава всички необходими данни и името на подпрограмата, в която са описани десните страни на системата от диференциални уравнения. На фигура 2.1.4 е показан фрагмент от кода на подпрограмата. Подпрограмата остава в безкраен цикъл (с етикет „**a:**“) за целия симулационен процес. Първото което се изпълнява е „чакане“ за настъпване на събитие с идентификатор (хандлер) **ha_beg**, което задава началото на изчисленията за всяка стъпка. Състоянието на това събитие се променя на подходящо място в родителската нишка (виж §2.1.3.6).

Следващият цикъл (с етикет „**b:**“) е свързан с вземане на подзадачи (системи от диференциални уравнения) за решаване от нишката. Чакането на събитието, свързано с хандлера „**loc_ha_1**“, е за синхронизация между нишките (race condition). Вземането на поредна задача е свързано с промяна на стойността на брояча **glb_counter**. При свършване на наличните подзадачи се излиза от този цикъл (с етикет „**b:**“) и се сигнализира родителската нишка, че тази именно нишка е свършила изчисленията, след което се отива в началото на безкрайния цикъл (с етикет „**a:**“). Тази подпрограма се обръща към главната подпрограма за числено интегриране **RKFASD**, където се извършва избор на метод.

```
SUBROUTINE     Integrator(th_id_num, num_threads,RHFun,adr,adr_len,numsat, &
                          thread_par,adr1,adr2,ha_1,adr_glb_count)
                  …
a: DO WHILE(.true.)
     k=  WaitForSingleObject(ha_beg,WAIT_INFINITE) ! an event for thread starting
b:   DO WHILE(glb_counter.LT. numsat)
     k=WaitForSingleObject(loc_ha_1,WAIT_INFINITE) ! ha_1 автоматично преминава
                                                   ! в "un-signaled" състояние докато се
        glb_counter= glb_counter + 1;        ! изчислява индекса на подзадачата която ще се
        loc_counter= glb_counter             ! обработва
                k= SetEvent(loc_ha_1)        ! промяната в състоянието на събитието позволява
                                             ! друга нишка да вземе подзадача за обработка
      IF(loc_counter.GT. numsat) EXIT
   CALL  rkfasd(…parameters…, RHFun ,  parameters  )
  END DO  b
   k= ResetEvent(ha_beg) ! подготвя събитието за стартиране на нишката при следваща стъпка
   k=    SetEvent(ha_end) ! сигнализиране за край за текуща стъпка по t, към родителска нишка
 END DO  a;

END SUBROUTINE  Integrator
```

Фигура 2.1.4.[4.1.8] Фрагмент от подпрограмата за управление на пул от нишки. Локалните променливи **loc_counter** и **loc_ha_1** съдържат стойностите на брояча на подзадачите и идентификатора на събитието **ha_1**.

2.1.3.5. Помощни подпрограми за работа с актуалните интегратори

За да се улесни подготовката на данните, които се предават чрез буферна програма към нишките на актуалните интегратори, необходимия за това код е обособен в две подпрограми **Data_AI** и **Preparation_AI** (допълнение **A**[3]). Всичко, което е необходимо, е да се извикат двете посочени подпрограми.

2.1.3.6. Синхронизация на изчислителните нишки с родителската нишка

Синхронизацията между всяка **ИН** на **АИ** и родителската нишка става с помощта на две събития с числови идентификатори (handler) **ha_beg** и **ha_end**. На определено място в родителската нишка (чрез подпрограмата **traekt_AI**) събитията **ha_beg** се поставят в сигнално състояние с което се инициират изчисленията от **АПИ** за текущата стъпка от времето, след което се чака **ИН** да завършат изчисленията и да сигнализират със събития **ha_end**. И в двата случая (и в родителската и в изчислителните нишки) се чака с функцията **WaitForSingleObject** (виж допълнение **A**).

2.1.3.7. Възможност за индивидуален модел на смущенията

За всеки обект, чието уравнение на движение се интегрира, може да се подбере модел на смущения според типа на орбитата, като по този начин се постига ефективност по отношение на времето за изчисления и точност на резултатите. Например, за по-голяма точност, при по-ниски орбити е необходимо моделът на гравитационното поле да включва по-високи хармоници. За илюстрация на ролята на различни хармоници на гравитационното поле е извършено прогнозиране (Atanassov, 2014b) на няколко движения по кръгови

[3] Допълнение **A** съдържа подпрограми на **ПИСОДУ** и може да се намери в основния текст на дисертацията.

орбити на различни височини. Всяка орбита е прогнозирана с различни модели на гравитационното поле, включващи нарастващ брой хармоници: 1x1, 3x3, 5x5, 10x10, 15x15 и 20x20. На различни височини приносът на различните смущения е различен за точното прогнозиране на движенията на спътниците.

Възможни са няколко варианта на модел на гравитационното поле. При използване на модел с разложение по сферични функции се посочва максималния брой на хармониците. При добавяне на смущения от страна на атмосферата е необходимо посочване на балистичен коефициент и маса на обекта. Може да се избира модел на атмосферата сред няколко – статична атмосфера, Jacia, CIRA, ГОСТ. На този етап са създадени възможности за избор на атмосферен модел, но добавяне на различни модели към системата за симулиране предстои. При по-високи орбити може да се добавят смущения от Луната и Слънцето.

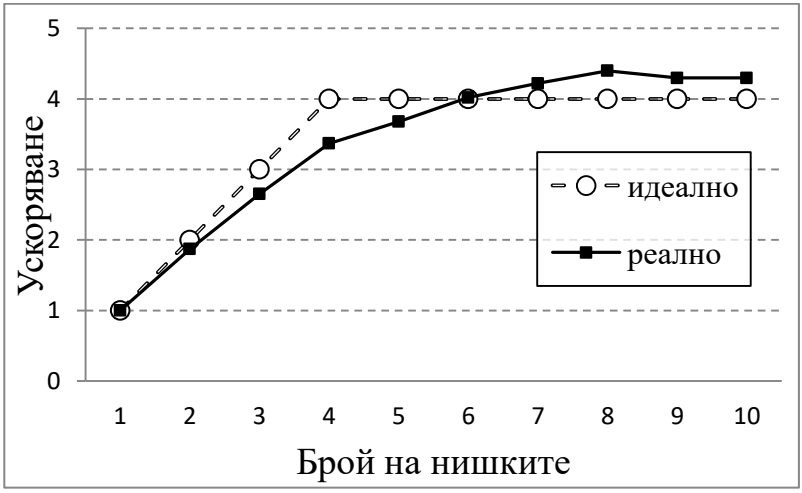
2.1.3.8. Оценка на ефективността на паралелния интегратор

Паралелният интегратор е разработен на основата на програмен език fortran 95 и библиотека DFMT под операционна система Windows. Числените експерименти са изпълнени с процесор Intel Core i7 2670QM.

За оценка на ефективността от прилагане на паралелни изчисления при прогнозиране на орбитални движения са проведени числени експерименти с интегриране на орбитите на 100 обекта (Atanassov, 2014). Симулациите са за период от един месец и са извършени с различен брой на изчислителните нишки – от 1 до 10.

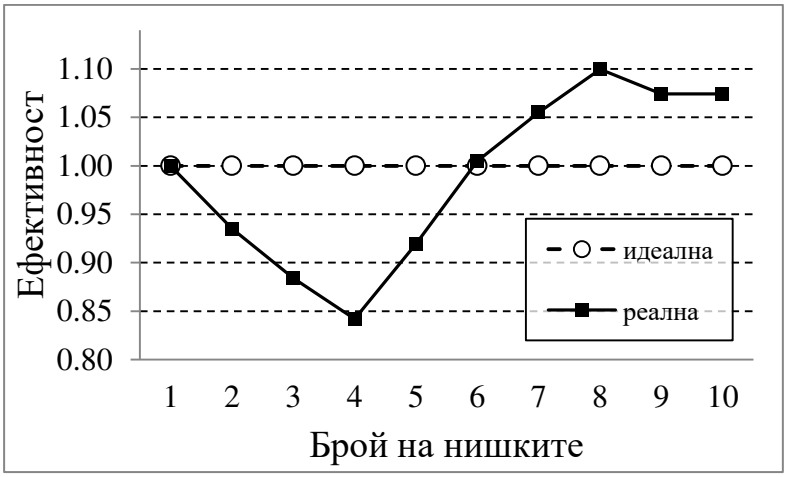
Ускоряването в следствие от паралелизацията е оценено съгласно формулата (виж §1.9.5):

$$S'_p = \frac{T_1}{T_p}$$



Фигура 2.1.5.[4.1.16] ■ – реално и ○ – идеално ускоряване на изчисленията в следствие от паралелизацията. Броят на процесорите е 4; при брой на нишките от една до четири ускоряването е сублинейно; увеличението на нишките над броя на процесорите увеличава ускорението и при повече от 7 става супер-линейно.

За разлика от формула (1) от §1.9.5, тук вместо времето за изпълнение на оптимален сериен код T_s е използвано времето $T_1 (T_s < T_1)$ при използване само на една нишка. В този случай **p** е броя на нишките. Тук по-скоро се оценява мащабируемостта (scalability) на системата компютър-алгоритъм. На фигура 2.1.5[4.1.15] са показани резултатите от проведения експеримент. Вижда се ефекта от възможността един процесор (ядро) да изпълнява две изчислителни нишки.



Фигура 2.1.6.[4.1.17.] Ефективност на паралелизирания интегратор в зависимост от броя на нишките. ■ – реална ефективност на алгоритъма при различен брой нишки, ○ – идеална ефективност в случай на липса на забавяния (overhead).

Аналогично, ефективността (фиг. 2.1.6) е оценена по модифицирани формули (Atanassov, 2016):

$$E'_p = \frac{T_1}{p.T_p}\text{, за } p \leq 4 \text{ и } E'_p = \frac{T_1}{4.T_p}\text{, за } p > 4\text{,}$$

където ***p*** е броя на нишките.

2.1.4. **Изводи относно паралелния интегратор**

Относно разработения паралелен интегратор на системи от обикновени диференциални уравнения може да се посочи следното:

- Изборът на оптимална схема за числено интегриране за всеки от разглежданите обекти при всяка стъпка от процеса на симулация, основан на използването на методи на РКФ с оценка на грешката, може да спестява изчислително време.
- С интегратора може да се прогнозира движението на голям брой обекти с различни параметри на орбитите, с постоянна стъпка във времето и поддържане на необходимата локална точност.
- Интегрирането с постоянна стъпка е важно от практическа гледна точка при проектиране на космически мисии.
- С избора за всеки обект на оптимален модел на смущенията, допълнително може да се повиши ефективността на изчисленията при решаване на задачи с

много обекти с различни параметри на орбитите (големи полуоси и ексцентрицитети).

– С интегратора могат да се симулират космически мисии и експерименти с много спътници и да се решават и други многомерни задачи.
– Разработеният интегратор може да се адаптира към решаване на различни типове задачи, както и да бъде вграждан в различни системи за симулации на космически мисии. Съпътстващите помощни подпрограми улесняват прилагането на интегратора и развитието на други приложения.

2.2. **Процесор за решаване на ситуационни задачи**

2.2.1. **Теоретични бележки**

За решаване на всяка научна или приложна задача свързана с провеждането на спътникови измервания или активни експерименти са необходими определени условия. Такива „благоприятни“ условия настъпват на определени участъци от орбитата – **орбитални събития**. Те се определят от специфични изисквания произтичащи от типа научни инструменти и окачването на спътниковата платформа, средата, обекта на изследване, пряка видимост, осветеност, ориентация на магнитното поле и други. Анализът за настъпването на едно орбитално събитие (**ситуационен анализ**) става чрез решаване на ситуационни задачи при които се проверяват различни ситуационни условия.

Обект на ситуационния анализ са спътници ведно с научните инструменти и служебни системи с които са оборудвани, тяхното движение, както и различни други обекти разположени върху Земята, небесната сфера или самата космическата среда с нейните компоненти, разглеждани в контекста на конкретна космическа мисия ангажирана с решаването на специфични задачи. Предмет на ситуационния анализ са търсенето и откриването на специфични орбитални събития от геометричен или физичен характер и определянето на времевите интервали в които те настъпват, когато е възможно или оптимално (с оглед на решаването на съответни задачи) извършването на конкретни спътникови операции. Търсенето на тези времеви интервали става въз основа на компютърни симулации, с използване на различни математически модели и методи.

В общия случай, една ситуационна задача **SP** може да се представи чрез логическа функция (Atanassov, 2008a):

$$1). \quad \mathrm{SP} = \mathrm{SP}\left(\vec{\mathrm{R}}, \{\alpha\}, \{\beta\}, t\right) = \begin{cases} false\ (\text{или } 0) \\ true\ \ (\text{или } 1) \end{cases}.$$

В (1) $\left\{\vec{\mathrm{R}}\right\} = \langle \vec{r}_1(t), \vec{r}_2(t), \ldots \vec{r}_n(t) \rangle$ са радиус-векторите на обектите в моделното пространство, $\{\alpha\}$ и $\{\beta\}$ – крайни множества от специфични ограничения и параметри съответно и t – време. Цялата ситуационна задача може да се изрази чрез независими ситуационни условия s_i свързани с различни ограничения относно:

Ситуационните условия са свързани с различни ограничения относно:

– **Условията на наблюдение**; оценка на възможни странични ефекти от смущаващи фактори – Слънце, Луна, радиационен фон;
– **Обекти на наблюдение**; те могат да бъдат (относително) статични или подвижни върху земната повърхност или небесната сфера;

– **Параметри на средата;** локални стойности за – атмосфера, йоносфера, магнитосфера.

Всяко ситуационно условие $\boldsymbol{sc}$ се дефинира чрез крайни множества от специфични ограничения $\{\alpha\}$ и параметри $\{\beta\}$:

$SP = sc_1(\{\alpha\}_1, \{\beta\}_1, t) \wedge sc_2(\{\alpha\}_2, \{\beta\}_2, t) \wedge \ldots \wedge sc_N(\{\alpha\}_N, \{\beta\}_N, t)$,

Намирането на орбитално събитие от гледна точка на ситуационния анализ означава за логическата функция ***SP*** да бъде изпълнено:

2). $SP = \boldsymbol{sc}_1(t) \wedge \boldsymbol{sc}_2(t) \wedge \ldots \wedge \boldsymbol{sc}_n(t) = true$

Множеството на предикатните функции описващи условията $\boldsymbol{sc}$ в (2) разглеждаме като наредено, което намира израз в начина по който се проверяват отделните ситуационни условия и се определя стойността на функцията ***SP***. Освен проверката на всички ситуационни условия, възможно е прилагането на правилото на Хорнер:

3). $SP = (\ldots(sc_1 \wedge sc_2) \wedge \ldots \wedge sc_{n-1}) \wedge sc_n$

При прилагане на това правило проверката се прекъсва при първото срещната неизпълнено условие, т.е. $SP = false$.

2.2.2. Програмна реализация на модела за представяне на ситуационни задачи

Всяко ситуационно условие се определя с определен брой параметри (с които се задава в най-простия случай някакъв геометричен модел) и съответни ограничения (ъгли, минимални или максимални стойности на величини свързани с решаваните задачи). Една ситуационна задача може да се основава на едно или повече независими ситуационни условия.

За дефиниране и решаване на голям брой ситуационни задачи, с повече от едно ситуационни условия, е необходимо тяхното гъвкаво и ефективно представяне в компютърната памет. Различните ситуационни условия се описват с различни по естество и брой параметри и ограничения (атрибути). За представяне в паметта се използва двумерен масив чиито елементи съдържат пълното описание на параметрите и ограниченията за всяко ситуационно условие. Всеки елемент представлява сложна структура от данни (потребителски тип), представена чрез основните типове (integer, real) или може да съдържа друга структура. За да може еднозначно да бъде интерпретиран всеки от елементите на този масив, всяко ситуационно условие съдържа идентификационен код, който се записва като цяло число от тип integer. Всеки стълб от този масив съдържа необходимите атрибути на отделните ситуационни условия, всяко от които се съдържа в отделен елемент.

Елементите от първия ред (нулев елемент) съдържат информация за ситуационната задача – брой на независимите ситуационни условия, флаг за изпълнение на задачата, номер на алгоритъм за оптимизация (1, 2 или 3) при повече от едно условие (0 ако оптимизация не се използва) и начало и край на времевия интервал в рамките на който всички ситуационни условия се изпълняват. Броят на редовете на масива е равен на максималния брой на ситуационните условия за всички ситуационни задачи, които се решават в рамките на текущия анализ. Броят на стълбовете е равен на броя на ситуационните задачи.

```
MODULE     RN
 type     SitCond
   integer        sit_code     ! код на ситуационното условие
   integer        sat_num      ! идентификационен код на спътника с който е свързано условието
   logical        flag         ! флаг за изпълнимост на условието: .false. или .true.
  union
   map    ! Sit__1: Преминаване над кръгова област с център (lati,longi) и ъглов радиус angle
    integer     reg_num          ! идентификационен номер на областта
    real         lati_r          ! ширина на центъра
    real        longi_r          ! дължина на центъра
    real        angle_r          ! ъглов радиус на областта
  end map
   map     ! Sit__2: видимост на спътник от наземна станция с координати (lati,longi)
    integer     grbstat          ! ground based stations numbers (0) and codes
    real           lati          ! ширина на станцията
    real          longi          ! дължина на станцията
    real          angle          ! ъгъл над хоризонта за спътника
   end map
  …! maps  за други ситуационни условия                                          (a)
  end union
 end type  SitCond
!_____________________________________________________________________
 type     sit_task
  union
   map                          ! параметри за цялата сит. задача
     integer     max_cond ! брой на ситуационните условия в ситуационната задача
     logical     flag          ! изпълнимост на ситуационната задача: .false. или .true.
     integer     opt_level   ! алгоритъм/евристика за оптимизация: 0 – без, 1/2/3
     real*8      t1,t2         ! съдържа последния времеви интервал, където задачата е изпълнима
  end map
  map
    type (SitCond) sit_cond     ! унаследяване на клас за ситуационните условия
  end map
  end union
 end type  sit_task      !                                                      (b)
END MODULE  RN
```

Фигура 2.2.1[4.2.1]**.** Модел за представяне на ситуационни условия. **(a)**. Моделът се състои от обща част и част съдържаща специфични за всяко ситуационно условие атрибути. **(b).** Моделът определя два варианта – единия за управляващи параметри за ситуационната задача (за $I^{-ви}$ елемент) и втория – за ситуационните условия.

Различните ситуационни условия се определят с различни по брой и смисъл параметри и ограничения. Параметрите ведно с ограниченията се описват чрез структура по начина показан на фигура 2.2.1 – вариант означен „**Sit__1**“.

Различните за всяко ситуационно условие атрибути са описани в програмен модул чрез структура с използване на езикова конструкция позволяваща алтернативно интерпретиране на всеки елемент от масива съдържащ описанията на различните ситуационни условия. Първите три атрибута (код на условието, номер на спътника и флаг за изпълнимост на условието) са общи за всички условия. Първите два от тях се задават при описание на задачата, а третият съдържа резултат от проверката на условието. Специфичните за всяко ситуационно условие атрибути се описват чрез оператора **union**, който позволява в зависимост от кода на условието да се използват различни формати при

записването на атрибутите и ограниченията в елементите на масива за съхранение, както и в последствие при тяхната интерпретация в хода на ситуационния анализ. Това е форма на динамичен полиформизъм по отношение на интерпретация на данни.

2.2.3. **Примери за разработени ситуационни условия**

- Преминаване на спътник над област от земната повърхност (определена като кръгов или правоъгълен сегмент);
- Преминаване през зоната на радиовидимост на наземна станция;
- Два спътника преминават „един над друг" в рамките на времеви интервал $\Delta\tau$ характеризиращ изследвано явление;
- Два спътника се движат „един над друг" над голяма част от земната повърхност, като разликата във времето е в рамките на времеви интервал $\Delta\tau$ характеризиращ изследвано явление;
- Спътник преминава над осветена/неосветена част от земната повърхност;
- Директна видимост между два спътника;
- Разстоянието между два спътника и ъгълът между векторите на скоростите им да са в определени граници;

2.2.4. **Оптимизация на ситуационния анализ основана на евристика с пренареждане на условията**

Всяко ситуационно условие sc_i се изпълнява в рамките на времеви интервал T_i и не се изпълнява в прилежащия му T_i^*. Единственото практическо условие е $T_i, T_i^* \gg \Delta t$. Задачата $\boldsymbol{SP}$ се решава за всяка стъпка Δt.

Вместо правата задача SP, за проверка на (2) или (3) (виж §2.2.1), нека да разгледаме обратната задача (Atanassov, 2008a):

4). $\overline{SP} = \overline{\boldsymbol{sc}_1(t) \wedge \boldsymbol{sc}_2(t) \ldots \boldsymbol{sc}_n(t)} = \begin{cases} true \\ false \end{cases}.$

Вместо (4), обратната задача можем да представим така (закон на де Морган):

5). $\overline{SP} = \overline{\boldsymbol{sc}_1(t)} \vee \overline{\boldsymbol{sc}_2(t)} \vee \ldots \vee \overline{\boldsymbol{sc}_k(t)} \ldots \vee \overline{\boldsymbol{sc}_n(t)}.$

За решаването на обратната задача е необходимо да е изпълнено само едно от условията $\overline{\boldsymbol{sc}_k(t)} = 0$. След откриване на k-то условие, достатъчно за решаване на задачата, може да се продължи с една от следните евристики (Atanassov, 2008a):

6). $\overline{SP} = \overline{\boldsymbol{sc}_k(t+\Delta t)} \vee \overline{\boldsymbol{sc}_1(t+\Delta t)} \vee \ldots \vee \overline{\boldsymbol{sc}_{k-1}(t+\Delta t)} \vee \overline{\boldsymbol{sc}_{k+1}(t+\Delta t)} \vee \ldots \vee \overline{\boldsymbol{sc}_n(t+\Delta t)}.$

Освен (6), може да се приложи още друга евристика (Atanassov, 2008a):

7). $\overline{SP} = \overline{\boldsymbol{sc}_k(t+\Delta t)} \vee \overline{\boldsymbol{sc}_{k+1}(t+\Delta t)} \vee \ldots \vee \overline{\boldsymbol{sc}_n(t+\Delta t)} \vee \overline{\boldsymbol{sc}_1(t+\Delta t)} \vee \ldots \vee \overline{\boldsymbol{sc}_{k-1}(t+\Delta t)}$

Изложеният подход, основан на пренареждане на условията за проверка, дава възможност за оптимизация на **CA**. Той беше приложен при управлението на занитен фотометър и спектрометър SATI.

За оценка на ефективността от прилагане на изложените евристики беше проведен числен експеримент, свързан с проверка на условията за провеждане на нощни измервания със спектрофотометрични инструменти. За период от една година бяха определяни положенията на Слънцето и Луната със стъпка от една минута по времето. В зависимост от подредбата на условията за Слънцето и Луната, бяха извършени съответно (412264, 525600, и общо 937864) и (525600, 171025, и общо 696625) проверки. С прилагане на (6) проверките бяха (468161, 186659, и общо 654820).

2.2.5. **Разработка на процесор за ситуационен анализ**

2.2.5.1. Сериен вариант

Серийният вариант на процесора за решаване на ситуационни задачи (**ПСА**) (Atanassov, 2013b) беше разработен като основна, управляваща подпрограма, която се обръща към подпрограми описващи моделите на съответните ситуационни условия.

2.2.5.2. Паралелен вариант на **П**роцесор за **С**итуационен **А**нализ

Можем да посочим няколко причини, поради които решаването на ситуационни задачи, изисква много изчислително време:

- Решават се голям брой ситуационни задачи;
- Решават се ситуационни задачи изискващи проверки на повече ситуационни условия;
- Проверяват се ситуационни условия изискващи значително изчислително време;
- Времевият хоризонт в рамките на който се провежда анализа е дълъг;
- Симулациите се повтарят многократно с различни параметри, което изисква повтаряне и на ситуационния анализ.

Както при паралелизация на изчисленията, свързани с числено интегриране на системи от диференциални уравнения, и тук могат да бъдат прилагани два подхода, аналогично на посочените от Gear, 1987:

- Паралелизъм в рамките на ситуационна задача (Parallelism across the situational problem);
- Паралелизъм в рамките на ситуационни задачи (Parallelism across the situational problems).

Изразът „паралелизъм в рамките на ситуационна задача“ изразява възможността отделни етапи от получаване на решението на една задача да са независими и да могат да бъдат изчислявани едновременно на различни процесори.

Изразът „паралелизъм в рамките на ситуационни задачи“ изразява възможност цели ситуационни задачи да бъдат решавани на различни процесори. Този подход е приложим при наличие на голям брой задачи за решаване.

Решаването на ситуационни задачи също е свързано с **нерегулярни изчисления**. Причините могат да са следните:

- Различни ситуационни задачи могат да включват различен брой ситуационни условия;

– Различни ситуационни задачи могат да включват различни по тип ситуационни условия;
– Различни (дори еднакви по тип) ситуационни условия могат да се изпълняват на различни части от спътниковите орбити (в различни времеви интервали).

На всяка стъпка от симулационното време се извършват различен брой проверки на ситуационните условия, поради прекъсването им в рамките на ситуационните задачи при срещане на неудовлетворено условие (3).

2.2.5.3. Използване на модела „пул от нишки“

Както при паралелния интегратор на **СОДУ** (**ПИСОДУ**), така и при процесора за **СА (ПСА)** е удобно да се приложи модела „пул от нишки“ за да се паралелизират изчисленията. Това е поради отнапред неизвестния брой ситуационни задачи за решаване и включените в тях ситуационни условия, водещи до нерегулярния характер на изчисленията.

Тук ще изложим общ подход за създаване и управление на актуални изчислителни инструменти, приложен към **ПИСОДУ** и **ППСА** (Atanassov, 2017). Този подход смятаме да бъде прилаган и към други паралелни инструменти, които предстоят да бъдат разработени.

2.2.5.4. Създаване на пул от нишки за **ППСА**

Създаването на пул от нишки става с подпрограмата **CreatePoolThreads,** която се използва при **ПИСОДУ** (виж §2.1.3.2). Предава ѝ се името на съответна буферна подпрограма, която ще се използва за приемане на данни от родителската нишка. Буферната подпрограма предава към подпрограмата за управление на пула освен приетите от родителската нишка данни и името на главната подпрограма от процесора за **СА**.

2.2.5.4.1. *Използване на полиморфизъм*

Новия подход се основава на **полиморфизъм –** възможност с едно име да се извикват различни подпрограми, според спецификата на списъците с актуални аргументи на всяка (Горелик, 2002; Akin, 2003). По този начин се постига използване на единствена подпрограма за управление на пулове от нишки за различни паралелни изчислителни инструменти (засега интегратор на **СОДУ** и процесор за **СА**). За тази цел се дефинират два потребителски типа **task_descriptor_template_integrator** и **task_descriptor_template_situations** (фиг. 2.2.3). Те са структури, съдържащи описанията на всички данни, необходими за изчислителните инструменти. Освен това, в конструкцията **interface** … **end interface** се описват имената на подпрограмите, които могат да се извикват с едно и също родово име (**UPC –** universal pool control). В случая са избрани две имена – **Pool_threads_Control_1** и **Pool_threads_Control_2**, които са синоними на една и съща подпрограма за управление на пулове от нишки, като второто е име на допълнителна входна точка, определена с оператор **entry** (фиг. 2.2.4[4.2.5]). Двете имена се различават по списъците с аргументи (фиг. 2.2.3 [4.2.5], 2.2.4[4.2.5]). Разликите се отнасят до предаваните имена на подпрограми – две при **ПИСОДУ** и една за **ППСА**.

Използването на буферна подпрограма е аналогично на това при **ПИСОДУ**. То се налага за предаване на данни от родителската нишка към нишките на **ППСА**. Буферната подпрограма съдържа името на главната подпрограма на **ПСА** и го предава на подпрограмата за управление на пула от нишки на **ППСА**. Фрагмент от буферната подпрограма **SituationProcessorUPC** е показана на фигура 2.2.3.б.

2.2.5.4.2. *Буферна подпрограма*

С оператора **USE** се дава достъп до съответните потребителски типове и интерфейси от модула **RN** (Допълнение **E**[4]). Структурната променлива **task_descriptor** унаследява променливата **task_descriptor_template_situations** от модула **RN** и съдържа всички данни, които са необходими за работата на **ППСА**. Променливата се въвежда за унифициране на подпрограмата за управление на пулове от нишки, за да се използва за управление на различни изчислителни инструменти (виж §2.2.5.4.3, фиг. 2.2.3.с).

2.2.5.4.3. *Подпрограма за управление на пулове от нишки*

Подпрограмата за управление на пулове от нишки е развита въз основа на предишни варианти, които служеха за управление на отделни изчислителни инструменти (**ПИСОДУ** и **ППСА**). Приложен е полиморфизъм по отношение на имената на подпрограмите за управление на двата разработени до сега инструмента. Дефиницията на този интерфейсен полиморфизъм е показана в фигура 2.2.3.а. По този начин може да се използва единствено името **UPC** независимо кой от двата изчислителни инструмента се извиква. С оглед на перспективите да се разработят и други инструменти, както и различните подпрограми за управление на пуловете от нишки за да се сведат до една единствена, въведени са две входно точки – едната **Pool_threads_Control_1** чрез оператора **SUBROUTINE** и втора **Pool_threads_Control_2** чрез оператор **ENTRY** (фиг. 2.2.5[4.2.5]). Това е направено поради необходимостта към различните инструменти да се предават различен брой имена на подпрограми.

Името **Solver** е име на главната подпрограма на изчислителен инструмент. То се съдържа в съответната буферна подпрограма. Името **RHFun** е допълнително за име на подпрограма, което се предава към конкретен инструмент. В случая на **ПИСОДУ** това е име на подпрограмата, в която са описани десните страни на **СОДУ**.

В подпрограмата **UPC**, за разлика от предишните варианти, се използва управляващ параметър **granule** (гранулация), който посочва колко подзадачи (системи от диференциални уравнения или ситуационни задачи) ще се решат „наведнъж“.

[4] Допълнение **E** съдържа дефиниции от модула **RH** и може да се намери в основния текст на дисертацията.

```
MODULE RN
…
 type        task_descriptor_template_integrator          b). user defined type containing
parameters
  integer      num_obj,num_equestions                              necessary  for integrator
  integer      adr1,adr2
  integer      adr_Grv_model,len_Grv_model
  character      izbor*1
 end type     task_descriptor_template_integrator
 !
 type        task_descriptor_template_situations     ! c). Потребителски тип съдържащ  данни
  integer      num_sat                                 !      и адреси за ситуационния процесор
  integer      t_adr,dt_adr,xvn_adr,xvk_adr
  integer      max_num_sit,num_sit_prob integer sci_problem_adr,len_sci_task
  integer      TrajectParam_adr,TrajectParam_len
 end type    task_descriptor_template_situations
 …
 INTERFACE UPC                              ! a). Дефиниция на интерфейсен полиморфизъм
  SUBROUTINE       Pool_threads_Control_1(th_id_num,num_threads,ha_1,adr_glb_counter, &

thread_par_local,lgranul,rkfasd_UPC,task_descriptor_adr,numobj,pertur)
    external      rkfasd_UPC,pertur
    integer th_id_num,num_threads,ha_1,adr_glb_counter,thread_par_local(4),lgranul
    integer          task_descriptor_adr,numobj
  END SUBROUTINE  Pool_threads_Control_1

  SUBROUTINE       Pool_threads_Control_2(th_id_num,num_threads,ha_1,adr_glb_counter, &

thread_par_local,lgranul,Psitanal_UPC,task_descriptor_adr,num_sit_prob)
    external       Psitanal_UPC
    integer        th_id_num,num_threads,ha_1,adr_glb_counter,thread_par_local(4),lgranul
    integer        task_descriptor_adr,num_sit_prob
  END SUBROUTINE Pool_threads_Control_2
 END INTERFACE UPC
 …
END MODULE RN
```

Фигура 2.2.3[4.2.3]**.** a) дефиниране на общо (родово) име **UPC**. b) и c) с тези структури се унифицират списъците с актуални аргументи при обръщение към подпрограмите за управление на пулове от нишки.

2.2.5.4.4. *Главна подпрограма на **ППСА***

Името на главната подпрограма на **ППСА** е обявено в съответната буферна подпрограма и се предава на подпрограмата за управление на пула от нишки. Всяка от нишките на актуалния **ППСА** извиква подпрограмата **Psitanal_UPC** (допълнение **B**[5]), като получава за решаване една ситуационна задача, за разлика от серийния вариант (виж §2.2.5.1). В цикъла с етикет **b** се проверяват ситуационните условия от задачата. От семантична гледна точка паралелния и серийния варианти не се различават съществено. Разликата е, че с цел унификация на управлението на пуловете от нишки, списъка от актуални/формални аргументи предавани на подпрограмата е сведен до брояч на

[5] Допълнение **B** съдържа подпрограми на **ППСА** и може да се намери в основния текст на дисертацията.

ситуационните задачи **loc_counter** и структурната променлива **task_descriptor**. Да припомним, че в буферната подпрограма се присвояват стойности на различните компоненти на тази структурна променлива. Тя съдържа началните адреси на различните места от паметта, където са разположени необходимите за решаване на ситуационните задачи данни. Достъпът до отделните елементи на масиви, съдържащи данни за конкретната ситуационна задача, става чрез явно изчисляване на адресите им (фиг. 2.2.5). За целта се използва оператора **POINTER,** чрез който се осъществява връзка между имената и изчислените адреси.

```
SUBROUTINE SatelliteIntegratorUPC(th_id_num)
 USE RN, only: perturb,UPC, task_descriptor_template_integrator                    a).
  external      pertur,rkfasd_UPC
  …
  type (task_descriptor_template_integrator) task_descriptor
  …
  task_descriptor%num_obj          = numobj;          task_descriptor%num_equestions= 6
  task_descriptor%adr_Grv_model= adr_Grv_model; task_descriptor%adr1              = adr1
  task_descriptor%len_Grv_model = len_Grv_model; task_descriptor%adr2             = adr2
…
  CALL UPC(th_id_num,num_threads,ha_1,adr_glb_counter,thread_par_local,lgranul, &
                    rkfasd_UPC,task_descriptor_adr,numobj,pertur)
END SUBROUTINE  SatelliteIntegratorUPC
   …
SUBROUTINE SituationProcessorUPC(th_id_num) !_situations
 USE RN, only:UPC, task_descriptor_template_situations                             b).
   external        Psitanal_UPC
   …
   type (task_descriptor_template_situations) task_descriptor
…
 task_descriptor%num_sat          = num_sat
 task_descriptor%t_adr            = t_adr;          task_descriptor% dt_adr        = dt_adr
 task_descriptor%xvn_adr          = xvn_adr;        task_descriptor%xvk_adr        = xvk_adr
 task_descriptor%max_num_sit  = max_num_sit;  task_descriptor%num_sit_prob = num_sit_prob
 task_descriptor%sci_problem_adr = sci_problem_adr;task_descriptor%len_sci_task= sci_task_len
 task_descriptor%TrajectParam_adr= TrajectParam_adr;
 task_descriptor%TrajectParam_len= TrajectParam_len;

 CALL UPC(th_id_num,num_threads,ha_1,adr_glb_counter,thread_par_local,granule, &
                    Psitanal_UPC,local_task_descriptor_adr,local_num_sit_prob)
END SUBROUTINE SituationProcessorUPC
```

Фигура 2.2.4[4.2.4]**.** Фрагменти от буферни подпрограми в които изчислителните инструменти се извикват с общо родово име **UPC**, но с различни списъци на актуалните аргументи. а). за **ПИСОДУ** и b). за **ПСА**. Специфичните за всеки инструмент аргументи са обединени в структури (без имената на подпрограмите).

2.2.5.4.5. *Подпрограми на ситуационните условия*

Различните ситуационни условия са описани в отделни подпрограми. Тук ще разгледаме ситуационното условие свързано с преминаване на спътник над кръгов сектор от земната повърхност и програмната му реализация. Кръговият сектор е определен с географските координати на центъра (φ,λ) и ъгловия му радиус θ (ъглов радиус – част от голям кръг, минаващ през центъра на сектора,

ограничена между този център и периферията на сектора). За разлика от други определения, това води към прилагането на формулите на сферичен триъгълник.

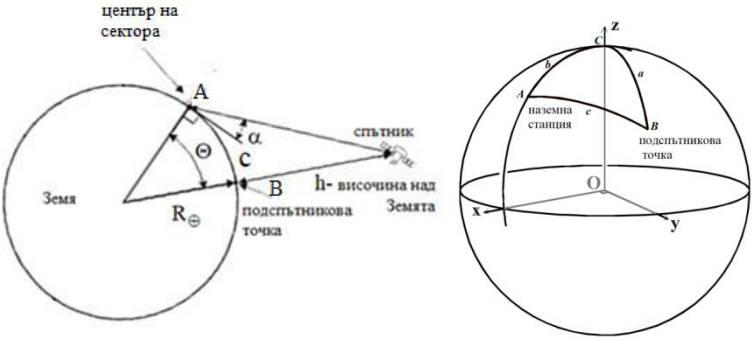
```
SUBROUTINE     Pool_threads_Control_1(th_id_num,num_threads,ha_1,adr_glb_counter, &
                    thread_par,granule,Solver,task_descriptor_adr,num_tasks,RHFun)
 USE DFmt
 USE RN
ENTRY          Pool_threads_Control_2(th_id_num,num_threads,ha_1,adr_glb_counter, &
                                  thread_par,granule,Solver,task_descriptor_adr,num_tasks )
  integer th_id_num, ha_1,adr_glb_counter,thread_par(4),granule
  integer task_descriptor_adr
  integer, automatic :: loc_ha_1,loc_counter,loc_counter0
  integer glb_counter
 POINTER(adr_glb_counter,glb_counter)

       ha_beg= thread_par(3); ha_end= thread_par(4)
      loc_ha_1= ha_1
a:DO WHILE(.true.);

          k= WaitForSingleObject(ha_beg,WAIT_INFINITE) ! Събитие за старт на нишката
 b:DO WHILE(glb_counter.LT. num_tasks)
    k= WaitForSingleObject(loc_ha_1,WAIT_INFINITE);
            obj_remain= num_tasks − glb_counter
      c:IF(obj_remain.GT.granule.AND.granule.GT.1) THEN
                loc_counte0= glb_counter+1; ! increment for serial subtask
                glb_counter = glb_counter + granule
                        k= SetEvent(loc_ha_1) ! Позволява други нишки да вземат задачи
         d:DO loc_counter=loc_counte0,loc_counte0+ granule-1
              IF(loc_counter.GT. num_tasks) EXIT
                        loc_adr= adr + (loc_counter-1)*adr_len;
                 CALL  Solver(loc_counter,task_descriptor_adr,num_tasks,th_id_num,RHFun)
            END DO d
                     ELSE
                 glb_counter= glb_counter + 1 ! брояч на взетите подзадачи
                 loc_counter= glb_counter;      ! запомня в локалната памет на нишката
                        k= SetEvent(loc_ha_1) ! Позволява други нишки да вземат задачи
              IF(loc_counter.GT. num_tasks) EXIT
                loc_adr= adr + (loc_counter-1)*adr_len;
               CALL  Solver(loc_counter,task_descriptor_adr,num_tasks,th_id_num,RHFun)
              ENDIF c
   END DO b
            k= ResetEvent(ha_beg) ! Подготовка на събитието за следваща стъпка по времето
            k= SetEvent(ha_end) ! Това събитие сигнализира за край на изчисленията
 END DO a;

END SUBROUTINE Pool_threads_Control_1
```

Фигура 2.2.5[4.2.5]**.** Унифицирана подпрограма за управление на пул от нишки за различни разработвани инструменти

2.2.5.4.5.1. Геометричен модел

На фигура 2.2.6.а[4.2.7.а] е показано разположението на центъра на кръговия сферичен сектор (означен с **A**) и подспътниковата точка (означена с **B**). Заедно със северния полюс (означен с C) трите точки определят сферичен

триъгълник. Страните b и a са известни, понеже са известни географските ширини на центъра на сектора и подспътниковата точка, т.е. $b = \pi - \varphi_A$ и $a = \pi - \varphi_B$. Освен това, $C = \lambda_B - \lambda_A$.



а). б).

Фигура 2.2.6[4.2.7]**.** а). Разполагане на центъра на сектора и спътник в равнина минаваща през центъра на Земята; б). разположение на станция и подспътниковата точка върху небесната сфера.

За сферическия триъгълник ABC (фиг. 2.2.6[4.2.7.b]) може да се приложи косинусовата формула (Бакулин, 1973, с. 28):

$$\cos c = \cos a \,.\, \cos b + \sin a \,.\, \sin b \,.\, \cos C \ .$$

За да се намира спътника над кръгов сферичен сектор, трябва частта от големия кръг, преминаваща през подспътниковата точка и центъра на сектора A (измервана с централен ъгъл с връх в центрърa на Земята – c^*) да е по-къса от ъгловия радиус на сектора, или:

$$\cos c \ > \cos c^*.$$

2.2.5.4.5.2. Програмна реализация

Важно е да се има предвид, че с цел да могат да бъдат проверявани едновременно ситуационни условия от един и същ тип, но от различни ситуационни задачи, локалните променливи на подпрограмите се дефинират като автоматични. Освен това, формалните аргументи свързани със ситуационното условие се записват в паметта отредена за ситуационната задача!

2.2.5.5. Помощни подпрограми

Аналогично на посочените в §2.1.3.5 помощни програми за инициализация и настройка на **ПИСОДУ,** са разработени и подпрограми за улесняване на работата с процесора за решаване на ситуационни задачи. Първата подпрограма е **Data_Sit_Solver**. С нея се подготвят за предаване към буферната подпрограма всички необходими за работата на процесора данни и адреси на променливи в които ще се записват резултатите. Тя спестява описанието на общи области в родителската нишка, които трябва да са идентични на тези в буферната подпрограма. Другата подпрограма е **Preparation_Sit_Solver**. С нея се осигурява старт на нишките на **ППСА**, както и преминаването им от суспендирано състояние, в което се оставят след създаването им, в работно при което могат да

„прочетат" заредените в общите области данни и да преминат към изпълнение – решаване на ситуационни задачи. Освен тези подпрограми, които се използват в началото, има още една подпрограма **sit_prob** (допълнение **B**) която се използва циклично на всяка стъпка от симулационния процес. С нея се стартират нишките на **ППСА** за съответната стъпка. Освен това, тя осигурява синхронизация между родителската нишка и нишките на **ППСА** (приложение **B**). След като се стартират нишките на **ППСА** се чака всички ситуационни задачи да бъдат решени, т.е. нишките да променят състоянията на събитията **ha_end** в несигнални.

2.2.6. Оценка на ефективността на паралелен процесор за ситуационен анализ

Оценка на ефективността на **ППСА** беше извършена на процесор Intel Core i7 2670QM. Този процесор има четири физически ядра, всяко от които може да изпълнява по две изчислителни нишки от инструкции.

Бяха извършени симулации за решаване на ситуационни задачи свързани с преминаване на един, два или три спътника с различни елементи на орбитите, над кръгов сегмент от земната повърхност. Като допълнително ситуационно условие беше използвано такова за проверка на осветеност на кръговия сегмент от Слънцето. Така бяха съставени 28 000 ситуационни задачи с едно, две, три или четири ситуационни условия. Тези задачи бяха решавани на всяка стъпка по времето след интегриране на движението на участващите спътници. Времената за изчисляване на ситуационните задачи бяха сумирани за период на симулиране 24 часа.

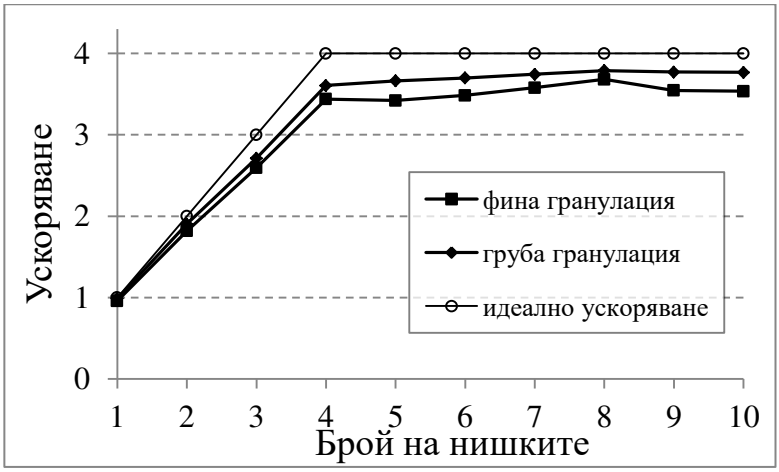
Оценките за ефекта от паралелизацията на ситуационния анализ бяха направени, както и по отношение на **ПИСОДУ** (виж §2.1.3.8), въз основа на популярните метрики ускоряване **S** и ефективност **E** (виж §1.9.5):

$$S_n = T_s/T_p$$
$$E_n = T_s/(n.T_p)$$

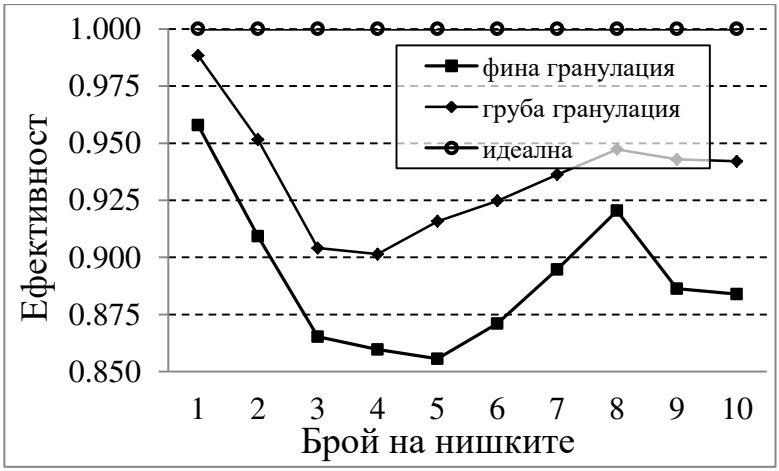
В последните формули T_s и T_p са съответно времената за серийно и паралелно изпълнения, със стартирани n изчислителни нишки. На фигура 2.2.7[4.2.12] е показано ускоряването в зависимост от броя на нишките при гранулации 1 и 5 (виж §2.2.5.4.3). Когато броя на нишките расте до достигане на броя на физическите ядра, ускоряването има линеен характер. При увеличаване на този брой за сметка на логическите ядра започват да се намесват различни допълнителни фактори (латентност, съгласуване), поради което нарастването на ускоряването макар да е на лице, може да не изглежда толкова линейно.

Аналогично, на фигура 2.2.8[4.2.13] е показана ефективността на процесора за ситуационни задачи в зависимост от броя на използваните нишки. Когато броят на нишките е по-голям от броя на процесорите, стойността на n остава равна на броя на процесорите. Когато броят на нишките е по-голям от броя на физическите ядра, но не по-голям от удвоения им брой, ефективността расте поради хипер-нишковата технология. Направените оценки показват малки разлики свързани с гранулацията. Тези разлики ще зависят от сложността на конкретните ситуационни условия и необходимите за тяхната проверка процесорни инструкции.

Благодарение на хипер-нишковостта, ходът на ефективността при брой на нишките по-голям от броя на ядрата (в случая 4) се обръща и расте. При брой на нишките по-голям от 8 ефективността отново намалява.



Фигура 2.2.7[4.2.12]**.** Ускоряване на изчисленията според броя на нишките при гранулации 1 и 5, означени съответно с '■' и '♦'. С '○' е означено идеалното ускоряване.



Фигура 2.2.8[4.2.13]**.** Ефективността от паралелизацията на изчисленията в зависимост от броя на нишките при гранулации 1 и 5, означени съответно с '■' и '♦'.

2.2.7. **Изводи относно ППСА**

Предлага се теоретично разглеждане на ситуационния анализ:

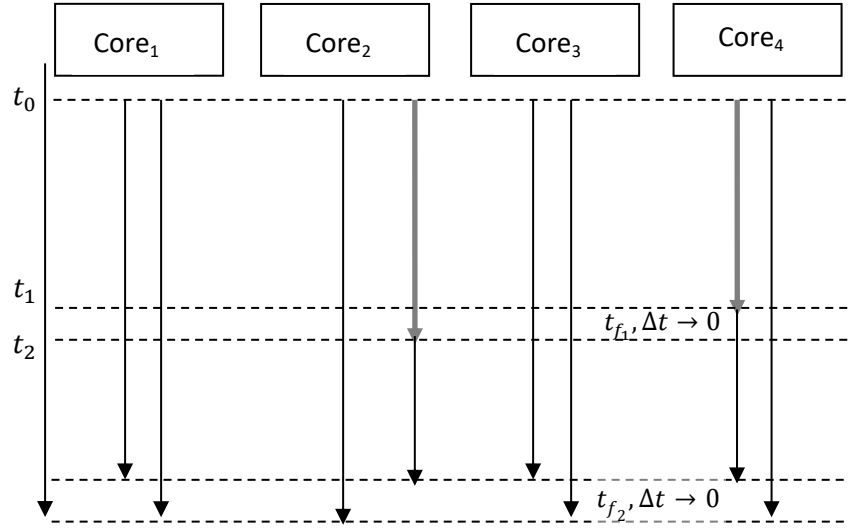
- Предлага се разглеждане на ситуационния анализ на основата на дискретната математика;
- Оптимизация на ситуационния анализ основана на евристики;

- Представен е програмен модел за представяне на ситуационни условия и ситуационни задачи; Разработени са модели за ситуационни условия и програмната им реализация;
- Разработен е паралелен програмен процесор за решаване на ситуационни задачи;
- Работата на **ППСА** е илюстрирана и оценена с числен експеримент; показан е ефекта от многонишковост;
- Разработени са класове, чрез които са представени разработените програмни изчислителни инструменти; прилагане на полиморфизъм;
- **ППСА** може да се развива като универсален инструмент за **СА** с възможност за прилагане при различни космически мисии с.

2.3. **Програмен модел „обединение** на **пулове** от **нишки**"

2.3.1. **Управление на паралелната работа на няколко актуални интегратора**

При симулация на системи с голяма размерност е възможно да бъдат стартирани повече от един актуални интегратора, които да работят едновременно при използване на наличните процесорни ядра. Необходимо е да се отбележи, че освен адаптивни и нерегулярни, задачите на различните актуални интегратори са с различен обем и изискват различно време за решаване. Възможно е последователно изпълнение на изчисленията за отделните интегратори. В този случай за ефективната работа на цялата система е необходимо:



Фигура 2.3.1[4.3.1]**.** Илюстрация на модела „**обединение на пулове от нишки**" за два пула. По-дебелите линии представят нишките на първия пул. В t_{f_1} и t_{f_2} нишките на двата пула приключват с изчисленията.

- Общият брой на нишките на всички интегратори да не е по-голям от броя на логическите процесорни ядра; тогава ще е налице излишно претоварване на системата.

– Броят на работещите нишки да не е по-малък от броя на логическите процесорни ядра; ако броят е по-малък, ще има неизползване на изчислителни ресурси.

За да бъдат удовлетворени горните изисквания е разработен програмен модел „**обединение на пулове от нишки**" (**ОПН**) (Atanassov, 1985a).

На фигура 2.3.1 е илюстрирано прилагането на модела. Разглеждат се два паралелни програмни инструмента, всеки реализиран въз основа на модела „пул от нишки". Единият инструмент (решава по-лесна задача) използва две нишки, а другият осем. Нишките на първият инструмент се изпълняват на втори и четвърти процесор. Шест от нишките на вторият инструмент заемат всички останали логически ядра. След като нишките на единият инструмент свършат своите изчисления, те стават неактивни и освобождават процесорни ядра. Неактивните две нишки на другия инструмент се активират.

„Обединение от пулове" е формален изчислителен модел, който може да бъде прилаган към различни паралелни програмни средства, основани на модела „пул от нишки". Отделните пулове могат да представляват както актуални версии на даден инструмент (интегратор, процесор за ситуационен анализ), така и различни инструменти.

2.3.2. **Програмна реализация на модела**

Моделът „обединение на пулове от нишки" е реализиран с програмния език fortran 95. Основава се на подпрограма за създаване на обединение и добавяне на пулове към него, както и на подпрограма за управление на работата на пуловете в рамките на обединението. След завършване на изчисленията, текущото обединение може да се унищожи. На следващ етап може да се създаде ново обединение, което съответства на следващ вариант на изчислителния модел, по който предстои да се извърши симулация.

2.3.2.1. Създаване на обединение от пулове

Създаването на „**обединение на пулове от нишки**" и добавянето на членове се извършва с подпрограма **CreateUnionPools**. При първо обръщение към нея се създава дескриптор на обединението. Адресът на дескриптора се пази в първия елемент на масива **union_atr**, който е последен в списъка на параметърите на подпрограмата. Във втория елемент се записва текущия брой на нишките на всички пулове. С всяко следващо обръщение към подпрограмата, към обединението се добавя следващ пул, като дескрипторът се допълва, а броят на нишките се коригира.

```
    …
CALL  CreateUnionPools(AI_2_thread_par,num_AI_2_threads,6, &
                                          AI_2_glb_counter,union_atr)
CALL  CreateUnionPools(AI_1_thread_par,num_AI_1_threads,2, &
                                          AI_1_glb_counter,union_atr)
    …
```

Фигура 2.3.2[4.3.2]. Илюстрация на създаване на модела „обединение на пулове от нишки" с включване на пуловете на два интегратора.

При обръщение към подпрограмата **CreateUnionPools** се предават основни параметри за конкретен пул, които се записват в дескриптора на обединението. Първият параметър **AI_2_thread_par** съдържа параметрите на пула, а вторият, **num_AI_2_threads** – броя на всичките му създадени нишки. Чрез третия параметър се определя броя на нишките които първоначално се активират. Следващият, четвърти параметър, предава адреса на брояча на пула. Последният параметър **union_atr** съдържа адреса на дескриптора, в който са записани различните атрибути на обединението.

2.3.2.2. Управление на работата на модела

Разработена е специална подпрограма **DynamicPoolsControl**, която е част от родителската нишка (допълнение **C**[6]). Тя управлява основните състояния на нишките на всички пулове, като се стреми да поддържа общия брой на активните нишки за работещите пулове да е максимален, равен на броя на процесорите в системата. Когато един изчислителен инструмент свърши всички задачи за изчисляване, нишките на неговия пул се деактивират. На тяхно място се активират нишки на друг изчислителен инструмент. Програмата **DynamicPoolsControl** е предназначена да бъде изпълнявана в цикъл по симулационното време (фиг. 2.3.3). Тя пази първоначалното състояние на създаде-

```
DO  t=T_start,T_final,step
        …
  CALL  DynamicPoolsControl(union_atr);
  CALL  Get_AI_rezults(num_sat,  t,dt,xvn,    xvk,  transfer_data1)
  CALL  Get_AI_rezults(num_sat_2,t,dt,xvn_2,  xvk_2,transfer_data2)
        …
END DO
```

Фигура 2.3.3[4.3.3]**.** Стартиране и управление на нишките от **ОПН** и вземане на резултатите за стъпка от симулационното време.

ното извън цикъла обединение от пулове за да може да го използва в началото на всяка стъпка по времето в хода на симулацията. Подпрограмата **Get_AI_rezults** се използва за прехвърляне на резултатите от работни масиви (ако е необходимо) в други, за по-нататъшно използване.

На този етап е извършен експеримент за установяване на работоспособността на модела с два **АПИ –** единия за симулиране на движението на шест спътника, а другия за движението на $30x10^3$ обекта.

2.3.3. **Дискусия**

Ползата от прилагане на модела **ОПН** е очевидна в случаи на повече от един актуални интегратора, с брой на нишките по-малък от логическите процесори. В този случай ще остават неизползвани изчислителни ресурси при последователно изпълнение на **АИ**. Този случай може да възниква по-често с увеличаване на броя на физическите ядра.

При обединяване на **АИ**, които решават задачи с различен обем изчисления, ползата от **ОПН** не е толкова очевидна и предстои да бъде

[6] Допълнение **С** съдържа подпрограми на **ОПН** и може да се намери в основния текст на дисертацията.

изследвана. В тези случаи се очаква по-добро използване на изчислителните ресурси при привършване на подзадачите, когато някои процесори първи приключват работата си и остават свободни.

Моделът позволява малки задачи вместо серийно, в рамките на родителска нишка, да се изпълняват чрез нишка, в рамките на обединение от пулове от нишки, заедно с други паралелни инструменти или други отделни нишки.

2.3.4. **Изводи относно модела „обединение на пулове“**

- Предложеният модел „**Обединение на пулове от нишки**“ представлява абстракция от по-високо ниво, от тази на модела „пул от нишки“;
- Моделът може да се окаже полезен при прилагане в динамични изчислителни сценарии;
- С увеличаването на броя на процесорите и ядрата в навлизащите изчислителни
- системи, може по-пълното използване на изчислителния им потенциал да стане проблем в определени случаи. Използването на предлагания модел може да е някакво решение в специфични ситуации.

2.4. **Алгоритми за планиране на спътникови операции**

Разглеждат се спътникови операции свързани с един общ безкраен ресурс и времеви прозорци за всяка една от заявените операции. Интересен е случаят когато времевите прозорци съвпадат частично. Количеството на заявките на различните операции предполагат конкуренция помежду им. Планирането на такива операции представлява известно предизвикателство. Хипотетично, можем да си представим пренос на данни между различни наземни станции чрез комуникационен спътник, преминаващ над тях.

2.4.1. **Модел за представяне на спътникови операции**

Предлагания подход за планиране на спътникови операции (**CO**) се основава на модел за тяхното описание. Този модел включва различни атрибути – някои от тях се използват за определяне на самите операции, други задават използването или неизползването на приоритети или други евристики в рамките на даден времеви интервал. Някои атрибути съдържат режима на изпълнение на задачите или самите резултати (фиг. 2.4.1).

Всяка спътникова операция има собствен идентификационен код **SO_idcode**, който я отличава от другите. Всяка **CO** може да бъде изпълнена в рамките на времеви прозорец, който се определя предварително с решаването на съответна ситуационна задача, посочена с идентификационния ѝ код **SP_idcode**. Заявеното време за изпълнение на някои операции се съдържа в атрибута **duration** и за някои се заявява от потребителя на услугата – за различните **CO** отнасящи се към различни научни (технологични) задачи е различно. В случая на комуникационни задачи заявената продължителност зависи от очаквано количеството на информацията, което може да зависи и от изпълнението на други задачи.

Атрибутът **priority** се използва за управление на реда на изпълнение на задачите. Задачите с по-висок приоритет се включват в търсенето на решението на задачата за планиране по-напред. Атрибута **regularity** позволява допълнителни възможности за управление на изпълнението на задачите. Например, една задача

може да бъде заявена за изпълнение еднократно в рамките на някакъв интервал от време (седмица, десетдневка, целия времеви хоризонт на анализа) или винаги, ако е възможно, както и до достигане на някакъв лимит по отношение на ресурс. Други варианти на този параметър могат да бъдат добавяни в бъдеще.

Атрибутите **t_begin** и **t_final** съдържат текущите резултати от планирането – моментите за начало и край за съответната операция. Атрибутите **counter** и **total_time** съдържат съответно броя на изпълненията и общото време за комуникация към всяка операция. Това представлява информация за хода на планирането, която може да се използва за оптимизация и подобряване на резултатите. Параметърът **color** се използва при визуализацията на процеса на планиране.

```
type       SO_model
 integer   SO_code  ! код на спътниковата операция
 integer   SP_code  ! SP(sit.prob.) код на ситуационната задача, която определя
                    ! времевия интервал са спътниковата операция
 real      duration ! вереме необходимо за спътниковата операция
 integer   regularity ! 0: еднократно изпълнение в интервала (date1 – date2)
                    ! 1- един път дневно; 2- един път седмично; 3- един път на месец
                    !-1: всеки път, когато ситуацията позволява
 real*8    interval(2) ! interval(1)- начален момент; interval(2)- краен момент (regularity=0)
                    !          (1)- release moment & due/deadline
 integer   priority  ! (0÷10)
 logical   fulfiled  ! .true. когато задачата е изпълнена и .false. когато не е
 logical   mark      ! има времеви интервал за планиране – .true.; няма – .false
 real*8    t_begin   ! планиран начален момент
 real*8    t_final   ! планиран краен момент
 integer   counter   ! брояч за колко пъти е планирана операция
 integer   unsched   ! брояч за непланиране на операция
 real*8    total_time ! общо планирано време за операция за целия интервал
 integer   color
end type   SO_model
```

Фигура 2.4.1[4.4.1]**.** Модел за представяне на задачите за планиране на спътникови операции

Моделът може да се развива като се добавят атрибути, описващи други характеристики на **СО**. В бъдеще може освен комуникационни, да се разгледат и други типове операции. При други типове операции някои от атрибутите могат да са едни и същи с тези на комуникационните, докато други ще отразяват техни специфики. Развитието на модела може да се осъществи на основата на полиморфно интерпретиране на определени части от паметта съдържащи различни по брой и семантика атрибути за различни типове **СО**, подобно на модела за описание на ситуационните условия (виж §2.2.2).

2.4.2. **Разработени алгоритми за планиране на спътникови операции**

Няколко алгоритъма за планиране са разработени в рамките на програмната система за симулации на космически мисии (Atanassov, 2015b). Те са варианти на известния „**greedy**" алгоритъм. Когато заявените задачи изискват повече от наличните за изпълнението им ресурси, тогава задачата за планирането им се

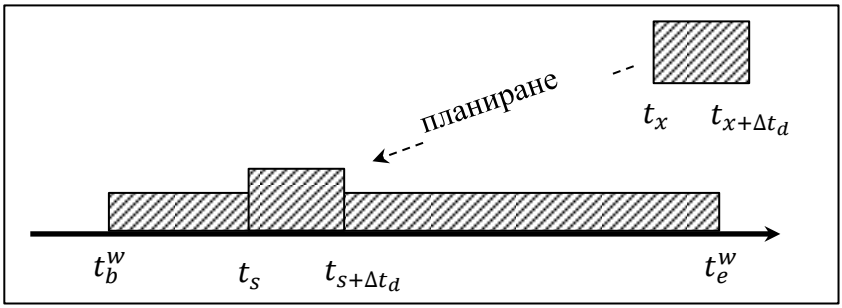
нарича „планиране на свръх заявени задачи". Когато имаме комуникационни задачи между спътник и наземни станции, при което времевите интервали в рамките на които отделните операции са възможни и се припокриват (частично), налице са свръх-заявени задачи. Спътникът преминава едновременно през радиозоните на няколко станции. Ситуационният анализ намира няколко повече или по-малко взаимно припокриващи се (частично) времеви прозореца, всеки от които съответства на преминаването през зоната на радиовидимост за една от станциите. Група от такива припокриващи се времеви прозорци, част от целия ред времеви прозорци за всички ситуационни задачи, ще наричаме фрагмент. Отделните последователни във времето фрагменти са разделени от времеви интервали в които няма условия за извършване на спътникови операции. Целият ред от решения на ситуационните задачи може да бъде представен като поредица от такива фрагменти и отделни времеви прозорци.

Целта на процеса на планиране е изготвяне на разписание за изпълнение на операциите в условията на различни ограничения (Pemberton & Galiber, 2001). На фигура 2.4.2 е илюстриран идеализиран модел на планиране.

Всяка спътникова операция се представя от (Pemberton & Galiber, 2001):

- Ситуационната задача, определяща разрешен времеви прозорец (t_b^w,t_e^w) – орбитално събитие;
- Времето (минимално) през което се изпълнява Δt_d.

Планирането представлява аналитичен процес, цел (може да не е единствена) на което е разполагане на операцията някъде в разрешения времеви прозорец, т.е. определяне на моментите за начало t_s и край t_f. Необходимо за това условие е $\Delta t_d \leq t_e^w - t_b^w$. Времето $\Delta t_s = t_e^w - t_b^w - \Delta t_d$ представлява луфт (slack), чийто размер може да играе съществена роля за изготвяне на времеви план (разписание), особено при много заявени задачи, чиито разрешени времеви прозорци се застъпват.



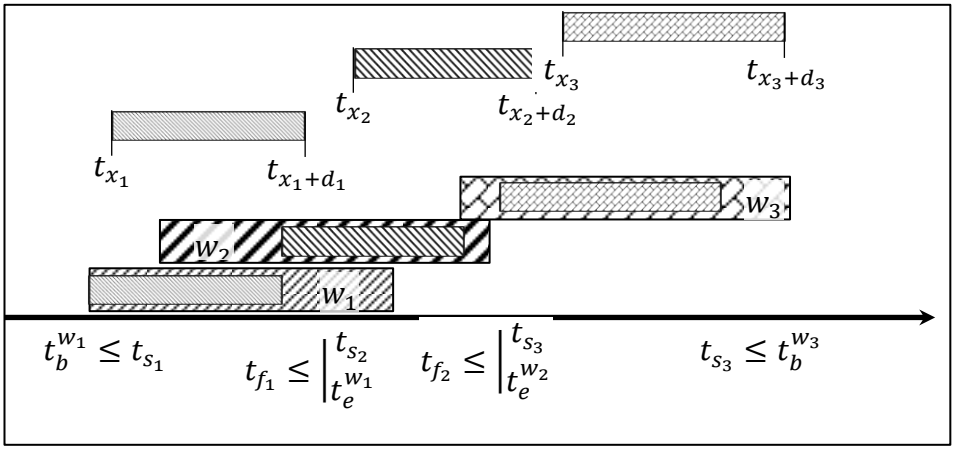
Фигура 2.4.2[4.4.2]**.** Илюстрация на процеса на планиране на спътникова операция.

На фигура 2.4.3 е показан пример с разполагане на три задачи в съответните им разрешени времеви прозорци. Планиране на една операция е възможно ако необходимото ѝ време **d** е по-малко от продължителността на времевия ѝ прозорец. Когато това е налице е добре да се разположи в началото на

прозореца. Когато времевите прозорци се застъпват, това скъсява времевите прозорци за следващите операции (при използване на един и същ ресурс).

Разрешените времеви прозорци $\left(t_b^{w_i}, t_e^{w_i}\right)$ се определят чрез **СА** и се подреждат в списък по моментите t_e^w, което е естествено. В §1.7.5 бяха посочени някои евристични правила на основата на които се разработват алгоритми за сравнения. Тук ще се спрем само на разработените алгоритми основани на възходяща подредба на моментите t_e^w ("first-finisheded first-scheduled" (**FFFS**)).

Водещи при анализа са разрешените времеви прозорци. Всеки от тези прозорци е определен с идентификатор на ситуационната задача, която го определя (допълнение **D**[7], подпрограма **sheduling**). Всички идентификатори се съдържат в масива **SP_code**. Началните, крайните моменти и продължителностите на времевите прозорци се съдържат в масивите **t1**, **t2** и **duration** с дължини **lenght_frag**. Всяка **СО** има атрибут, който съдържа идентификатора на ситуационната задача, която определя разрешените времеви прозорци (виж по-горе модела за **СО**). Последователно се разглежда всеки времеви прозорец и се разполага съответната **СО**, като началото ѝ се поставя по възможност в началото на времевия прозорец. След това, интервалът от време в който се назначава текущата **СО** се изрязва от всички разрешени времеви интервали $(t_b^{w_i}, t_e^{w_i})$ с които се припокрива. В резултат, получените нови стойности на разрешените времеви прозорци се проверяват дали продължителността им е достатъчна за извършване на съответните **СО**, и ако не се отстраняват от по-нататъшно разглеждане. Масивът **Task** съдържа всички операции, които предстои да се планират. Резултатите от планирането на **СО** за един фрагмент с разрешени времеви интервали се съдържат в масива **schedule**.



Фигура 2.4.3[4.4.3]**.** Пример за планиране на три операции в рамките на разрешените им времеви прозорци.

[7] Допълнение **D** съдържа подпрограми свързани с **ПСО** и може да се намери в основния текст на дисертацията.

Друг пример за разработка на средства за планиране е показан в допълнение **D**. Подпрограмата **Scheduling_2** извършва планиране на операции в зависимост от назначени приоритети. Подпрограмата **select_Task_by_priority** извлича последователно всички операции с даден приоритет и ги анализира. След разполагане на дадена операция в разрешения ѝ от **СА** времеви прозорец се прави преразглеждане на параметрите на други времеви прозорци, с които текущият се застъпва.

2.4.3. Илюстрация на планиране на спътникови операции

Показан е графичен фрагмент съдържащ времевите прозорци на фрагменти от **СО** и отделни стъпки от процеса на планиране, на основата на използвания алгоритъм. За целта е разработена програма, която визуализира всяка стъпка на процеса на планиране.

2.4.4. Изводи относно разработените алгоритми и програми за ПСО

- Основната цел на разработените алгоритми за **ПСО** е да демонстрират връзката и значението на **СА** за подготовката и управлението на космическите мисии;
- Разглеждането на **ПСО** още на ранни етапи от подготовката на космически мисии дава информация относно параметрите на специфични за конкретните мисии ресурси и позволяват да бъдат оптимизирани.

2.5. Програмна система за симулации на космически мисии и експерименти

Разработваната програмна система за проектиране и симулации на космически експерименти има експериментален характер. Основното ѝ предназначение, на този етап, е да служи за развитие и експериментиране с разработваните изчислителни инструменти. Системата включва основни инструменти (виж §1.3) и съпътстващи помощни средства, попадащи в различни области. На този етап в системата са включени двата инструмента **ПИСОДУ** и **ППСА**.

2.5.1. Структура на системата

Системата представлява съвкупност от две подсистеми – диалогова, за управление и въвеждане на параметрите на изчислителните модели и изчислителна, съдържаща разработваните изчислителни инструменти. Взаимодействието между двете подсистеми става със събития и глобални променливи. Със събитията се предават сигнали към различни точки от програмата, а чрез глобалните променливи се предават стойности за управление на работата на програмата.

2.5.2. Диалогова подсистема

2.5.2.1. Падащи менюта

Чрез падащите менюта се активират различни части на диалоговата подсистема за въвеждане на различни варианти на модели за симулации – орбитални параметри, модели на смущенията, дефинират се ситуационни задачи. Има и менюта с които се избират различни режими на работа на системата. Всички показани форми са експериментални, показани са за илюстрация и в бъдеще могат да претърпят промени.

Най-основната функция на диалоговата подсистема е да създава нови проекти и да избира по кой от вече създадените ще се работи. След избор на проект и въвеждане на симулационен модел (на този етап може да включват модели на движение и ситуационни задачи) се активират и други падащи полета. В хода на симулацията може да се промени (ускори, забави или приравни към „реално време“) скоростта с която се визуализират резултатите от всяка стъпка по времето. Може изчисленията да се спират, прекратяват или да се повтарят.

С помощта на полетата от падащото меню се извършват различни настройки. Например, могат да се посочат броя на нишките на актуален интегратор и процесора за ситуационен анализ (**ПСА**), стъпка по времето.

2.5.2.2. Въвеждане на модели на движение

Диалоговата подсистема съдържа част за въвеждане на модели на движение за обектите от нова космическа мисия и тяхното развитие в последствие (добавяне на нови варианти или промени на вече въведени). Използва се диалогова кутия с множество контроли за въвеждане на параметрите на модела. Освен орбиталните елементи, за всеки спътник отделно може да се определят смущаващите движението сили. На този етап са включени шест варианта на гравитационно поле:

- Централна сила,
- Нормално гравитационно поле,
- Първи шест зонални хармоници (Эскобал, 1970),
- Зонални хармоници $20^{-ти}$ ред (Прохоренко, 1976),
- Смесени хармоници до $20^{-ти}$ ред (Прохоренко, 1976).

Включени са диалогови контроли за въвеждане на смущения от атмосферата (избор на атмосферен модел и параметри на спътника (§2.5.2.2)) и смущения от Луната и слънцето (§2.5.2.3).

По подразбиране се включва модел с централни сили. С помощта на различни бутони може да се избира друг модел на гравитационните сили. Веднъж активирани, контролите за определяне на модела на силите за даден спътник остават и за следващи спътници, докато параметрите не се променят или силите не се изключат със съответните контроли.

Въвеждането на различните параметри на модела на движение е защитено срещу грешки. За всеки от параметрите се прави специфичен семантичен анализ на въведената стойност и при несъответствие в съответното поле се извежда съобщение за грешка. При въвеждане на орбитални параметри (голяма полуос и ексцентрицитет) се проверява дали орбитата е възможна.

Въведените модели за движение на спътниците могат да се запомнят като отделен вариант и по-късно да се използват.

2.5.2.3. Диалогов редактор на ситуационни задачи

Въпреки че **ПССКМЕ** се разработва с експериментална цел и за тестване на разработваните изчислителни инструменти, разгледани по-горе, специално внимание се отделя на ситуационния анализ. За да може **ПСА** да бъде прилаган е необходимо да са налице предварително дефинирани ситуационни задачи. На етапа на начално проектиране и анализ на космически мисии е важно лесно и удобно да се съставят ситуационни задачи на основата на различни типове

ситуационни условия. При мисии с повече спътници и различни инструменти може да се анализират възможностите за решаване на разнообразни научни задачи. Всяка научна задача може да породи една или повече ситуационни задачи. Някои ситуационни задачи могат да се различават по параметрите и ограниченията в ситуационните условия които включват. Естествено е в хода на анализите да се обсъждат различни варианти на множествата от задачи. Затова е разработен вариант на диалогов редактор за съставяне на ситуационни задачи.

Диалоговият редактор представлява форма с различни диалогови контроли за посочване на ситуационни условия и въвеждане на определящите ги параметри и ограничения. Различните полета за въвеждане на параметрите и ограниченията, както и съответните им имена се появяват контекстуално, в зависимост от избраното ситуационно условие. При композиране на ситуационна задача с повече от едно ситуационни условия, редът им може да бъде променян с използване на бутони. Въведено ситуационно условие може да се изключи преди ситуационната задача да бъде завършена.

Сценариите за работа с диалоговия редактор се определят чрез използваните командни бутони. Веднъж въведени, стойностите на параметрите и ограниченията на дадено ситуационно условие или ситуационна задача с повече условия, те могат да бъде запазени и по-късно използвани. Могат да бъдат запазвани и всички текущо въведени ситуационни задачи. При следващ сеанс, запазени задачи могат да се разглеждат и евентуално използват. Предвидени са възможности за посочване на методи за оптимизация на **СА**.

2.5.3. **Изчислителна подсистема**

Изчислителната подсистема е организирана като стандартна програма – има главна програма, която играе ролята на автомат, който получава и интерпретира управляващи сигнали и съответни глобални променливи, с което се преминава към изпълнение на съответен изчислителен модул:

Главната програма на **ПССКМЕ** (допълнение **Е**[8]) съдържа безкраен цикъл, в който се чака за настъпване на събитие с идентификатор **ha_0**, с което се сигнализира за наличие на нова команда. Командите се съдържат в текстовата променлива **menu**, стойността на която се предава чрез обща област с име **/cBasicControl/**. При първото преминаване през цикъла се модифицира основното меню (фиг. 4.2). След това с команди от менюто се манипулират променливата **Menu** и събитието с идентификатор **ha_0**. Чрез конструкцията **SELECT CASE … END SELECT** се интерпретират формираните команди и се стартират основни изчислителни процеси. Този подход позволява в лесно развитие на изчислителните възможности на **ПССКМЕ**.

Самия симулационен процес се управлява от подпрограмата **Driver**. На този етап възможностите за избор на изчислителен сценарий се основават на стойностите на елементите на логическия масив **TypeTask**. В §1.3 са посочени основни типове изчисления, каквито са предвидени и в **ПССКМЕ**. Стойността на

[8] Допълнение **Е** съдържа подпрограми свързани с **ПССКМЕ** и може да се намери в основния текст на дисертацията.

първия елемент е винаги ‘**.true.**’, докато другите се определят в зависимост от изчислителните задачи, които ще се изпълняват.

Хода на симулационното време се управлява в цикъла с конструкцията „**DO** … **END DO**”. Конструкциите в началото на цикъла, с етикети '**b**' и '**c**', и в края с етикет '**j**', както и функциите за синхронизация забързват, забавят или приравняват симулационното време към реалното.

Подпрограмата **traekt_AI** стартира интегратора на уравненията на движение (виж §2.1.3.6). Тук е показан вариант в който масивите сред списъка на актуалните аргументи се представят с адресите си. След като координатите на спътниците бъдат определени за текущ момент от време, при необходимост могат да бъдат определени множество геометрични и физически параметри съответстващи на положението на всеки спътник по орбитата:

- Географски координати на под-спътниковата точка, височина на спътника над земната повърхност,
- Компонентите и модула на вектора на магнитната индукция, компоненти на средата (атмосфера, йоносфера, магнитосфера), температура, налягане.

Конструкцията с етикет '**e**' управлява работата на **ППСА** за всяка стъпка от симулационното време.

Записът и визуализацията на резултати от изчисленията се управлява от конструкциите с етикети '**h**' и '**i**'. На този етап визуализацията е развита в по-голяма степен и се отнася към:

- Представяне на следите на космически обекти върху двумерно изображение на земната повърхност;
- Онагледяване на някои ситуационни задачи върху изображение на земната повърхност, звукова индикация;
- Визуализация на времевите интервали, определени от **СА**.

2.5.4. **Планиране на спътникови операции**

За планиране на **СО** са разработени няколко подпрограми (виж §2.4). Те се стартират от полето на главния списък „scheduling“. На този етап тези програми са включени с експериментална цел, за илюстрация на резултатите от ситуационния анализ. Връзката между **СА** и планирането на **СО** е чрез файл, който съдържа резултатите от **СА**.

Освен основните, разработени са още помощни подпрограми, за въвеждане на атрибути на заявените операции и за илюстрация на процеса на планиране. Чрез подходяща визуализация се показва динамиката на процеса на планиране.

За онагледяване на процеса на планиране са разработени допълнителни подпрограми с които се постига по-ясна представа за самия процес на разполагане на задачите във времето, както и крайния резултат.

2.5.5. **Изводи относно разработваната система**

На този етап разработваната програмна система има експериментален характер и основно е предназначена за разработка и тестване на изчислителни инструменти и модели. Бъдеща работа би довела до създаване на практически приложима програмна система за анализ и проектиране на космически мисии.

Характерни особености на системата:

– Включване на различни, основни изчислителни инструменти основани на ефективни методи, предназначени за решаване на задачи с големи размерности;
– Ефективни средства за диалог за съставяне на гъвкави сценарии за симулации;
– Средства за контрол и управление на изчисленията;
– Многофункционалност – решаването на динамични задачи свързани с движение на спътници е само основа за решаване на различни други задачи;
– Възможности за развитие и прилагане към различни спътникови мисии.

Глава III. Приложение на разработваната програмна система

ПССКМЕ е използвана на различни етапи от нейното развитие, основно при разработката и изследването на отделните изчислителни инструменти, разгледани в **глава II**:

– Паралелен интегратор (**ПИСОДУ**),
– **ППСА**, както при разработка и тестване на различни ситуационни условия;
– Програмен модел **ОПН** за съвместно използване на изчислителните ресурси от инструменти основани на модела „пул от нишки“ (виж §2.4);
– Разработка на алгоритми за планиране на **СО**;
– Разработка на проблеми не включени в текста на дисертацията.

За получаване на цялостно впечатление от прилагане на **ПССКМЕ** се извърши частичен анализ на мисия свързана със спътник на слънчево-синхронна орбита. Спътникът преминава над изследователски станции разположени на Антарктида, както и над територията на България. Анализира се възможността за комуникация между спътника и станциите за пренос на информация. Беше приложена разработрваната **ПССКМЕ** за предварителен анализ на преминаването на спътника над станциите за обмен на информация (Atanassov, 2015b).

3.1. Задача за решаване

Разглежда се спътник с основни оптични инструменти, способни да снемат изображения на земната повърхност. С оглед на основните задачи свързани с изображения, като подходяща за целта се разглежда слънчевосинхронна орбита с височина 800 км и наклонение 84°. Като допълнителна възможност се обсъжда възможността за обмен на информация (изследователска, медицинска, от битов характер) между спътника и станциите, и пренос към център за управление.

Зоната на радио-видимост за всяка станция зависи от височината на спътника над земната повърхност, географските координати на спътника и изискване за спътника да бъде на някакъв ъгъл над хоризонта. Разглеждани са различни случаи на ъгъл над хоризонта между 15° и 30°.

3.2. Стъпки от прилагане на **ПССКМЕ**

а). *Създаване на нова мисия*
Създаването на нова мисия позволява различни входни данни (орбитални параметри и модели, както и ситуационни задачи и резултати) да се обвържат с анализи относно конкретния проект.
б). *Въвеждане на орбитални параметри и модели*

Бяха въведени различни орбитални параметри (отговарящи на слънчево-синхронни орбити) и модел на гравитационни смущения.

в). *Въвеждане на ситуационни задачи*
Бяха въведени ситуационни задачи свързани с преминаване на спътник през зоната на радиовидимост за всяка от разглежданите 72 станции на Антарктида. Бяха разглеждани варианти с 15°, 20°, 25° и 30° над хоризонта.

г). *Проиграване на различните варианти на модела*
Бяха получени различни варианти на търсените **времеви интервали**, подходящи за комуникация на всички станции със спътника.

д). *Проиграване на планиране на операции*
Бяха проиграни различни варианти на планиране на спътникови операции с избран вариант на резултати от ситуационния анализ и спътникови операции с различни заявени времена за комуникация, както и планиране с назначаване на приоритети за спътниковите операции.

3.3. Резултати от прилагане на ПССКМЕ

Беше установено, че за различни слънчево-синхронни орбити, с каквито беше експериментирано, продължителността на един фрагмент от застъпващи се времеви прозорци (§2.4.2) е между 15 и 20 минути (съответства на географското разположение на станциите), докато преминаването на спътника над зона на радиовидимост на център за управление разположен на територията на България е в рамките на десетина минути. Освен това, спътникът преминава над станции при всяка своя обиколка, докато преминаване над територията на България е при няколко обиколки в денонощие, групирани в две съседни по две или три (в зависимост от изисквания за ъгъл над хоризонта на спътника и изискване за минимална продължителност на сеанс). Тези резултати са тривиални, но важното е, че системата ги предоставя за по-сериозни анализи.

Резултатите показват, че географското разположение на станциите е съществено при планирането на операциите за всяка от тях. Резултатите от планирането не се влияят съществено както от основните параметри на орбитата на спътника, така и от продължителността на комуникационните операции. Въпреки че задаването на по-малка продължителност позволява на повече станции да комуникират със спътника, поради разположението си, някои от станциите имат предимство.

Назначаването на приоритети беше правено с генератор на случайни числа. Резултатите не показаха съществена разлика спрямо предишните резултати, което показва, че ролята на географското разположение на станциите е основна. Разработката и прилагането на специални евристики (като посочената в §2.4.1, включена в модела за **СО** на фиг. 2.4.1) е въпрос на бъдеща работа.

3.4. Изводи относно прилагането на ПССКМЕ

– С разработваната програмна система беше изпълнен пълен цикъл за съставяне на модели за симулация на орбитално движение на спътник, ситуационен анализ за установяване на времеви интервали за преминаване през зоните на радиовидимост на наземни станции и планиране на спътникови операции за обмен на данни. Получените резултати демонстрират работоспособност на **ПССКМЕ**.

– Въпреки експерименталния характер на разработваната система (основната ѝ цел е да се подпомогне разработката и тестването на изчислителните инструменти), тя предоставя възможност за получаване на резултати при проектиране на космически мисии и представлява основа за по-нататъшна работа.
– Разработваната система се оказва ефективна както по отношение на възможността за въвеждане на модели за решаваните задачи (на движение и ситуационни задачи), така и по отношение на ефективността на тяхното решаване.

IV. Заключение и бъдеща работа

С настоящата работа се поставя основа за развитие в областта на проектирането на космически мисии и по-специално на ситуационния анализ. По-нататъшните усилия могат да бъдат съсредоточни в следните направления:

– Интегратора на системи от диференциални уравнения може да бъде развиван чрез включване на други методи за интегриране – едностъпкови, оптимизирани методи **RKF** от по-висок ред (Dormand and Prince, Verner, Tsitouras), Bulirsch-Stoer, методи за решаване на уравнения от втори ред (Nyström, Fehlberg), метод на Еверхард, многостъпкови методи (Adams-Bashforth-Moulton, Störmer-Cowell, Gauss-Jackson). Включването на интерполанти е по-важно от включване на по-точни методи и схеми;
– Процесора за решаване на ситуационни задачи може да се развива чрез разработка на алгоритми и подпрограми за ситуационни условия, както и оптимизиране на процеса на решение (евристични методи за оптимизация, решение на задачи с общи ситуационни условия);
– Изследване на влиянието на размерността на задачите решавани от **ПИСОДУ** и **ПСА**, степента на гранулация на целите задачи и локалността на данните участващи в изчисленията, върху ускоряването и ефективността от паралелизацията на изчисленията;
– Включване на разработвани модули за симулация и визуализация (астрономичен, инжекция на заредени частици);
– Включване на модели за изчисляване на параметри на околоземната среда (атмосферни, йоносферни и магнитосферни модели);
– Развитие на модела „обединение на пулове от нишки“;
– Развитие на език за синтез на комплексни модели и интерпретатор;
– Разработка на други изчислителни инструменти (функционален анализ на разпределени спътникови системи).

V. Приноси на дисертационния труд

Научни приноси:

- Разработена е стратегия за избор на метод с оптимален ред, за интегриране на уравнение на движение на спътници (Атанасов, 1986, 1987, 2013a); предложени са евристики за оптимизация на ситруационния анализ (Atanassov, 2008a);
- Разработени са абстрактни класове и методи за съставяне на модели на движения, ситуационни задачи и различни ситуационни условия (Atanassov, 2016, 2017);
- Разработени са паралелни изчислителни инструменти – **П**аралелен **И**нтегратор на **С**истеми от **О**бикновени **Д**иференциални **У**равнения (**ПИСОДУ**) (Atanassov, 2007, 2014b) и **П**аралелен **П**роцесор за **С**итуационен **А**нализ (**ППСА**) (Atanassov, 2013b, 2014a, 2016);
- Разработен е програмен модел **О**бединение на **П**улове от **Н**ишки (**ОПН**) (Atanassov, 2015a)
- Разработени са основите (методология) на **П**рограмна **С**истема за **С**имулиране на **К**осмически **М**исии и **Е**ксперименти (**ПССКМЕ**) (Atanassov, 2013b).

Научно-приложни приноси:

- Създадени са паралелни програмни инструменти **ПИСОДУ** и **ППСА** с възможности за вграждане в други програмни системи;
- Създадения програмен модел „**обединение на пулове от нишни“** може да бъде прилаган в други програмни системи;
- Създадените алгоритми за **ПСО** могат да бъдат практически използвани;
- **П**рограмната **С**истема за **С**имулиране на **К**осмически **М**исии и **Е**ксперименти се проектира и създава за да бъде използвана.

VI. Списък на публикациите по темата на дисертацията

1. **Атанасов**, Ат., 1986, Программная реализация алгоритма определения оптимальной схемы численного интегрирования, систем дифференциальных уравнений космической навигации. Сб. *8-ой конференции Секции № 6 „Информационное обеспечение космических експериментов“*, Стара Загора, 19-25.10.1986, сс. 7 – 12.
2. **Атанасов**, Ат., 1987, Начин за повишаване на ефективността при числено интегриране на обикновени диференциални уравнения с постоянна стъпка, за нуждите на имитационното моделиране. Сб. "*Математика и математическо образование*", София, БАН, сс. 306 – 309.
3. **Atanassov**, A.M., 2007, Integration of the Equation of the artificial Earth's Satellites Motion with Selection of Runge-Kutta-Fehlberg Schemes of Optimum Precision Order, *Aerospace Research in Bulgaria*, No. 21, pp. 24 – 34.
4. **Atanassov**, A.M., 2008a, Enhancing the Efficiency in Checking Constraints Satisfaction when Planning Ground-based and Space Experiments, Using an Alternative Problem, *Aerospace Research in Bulgaria*, 22, pp. 51 – 58.
5. **Atanassov**, A.M., 2008b, An Adaptive Parallel Integrator of Ordinary Differential Equations System for Space Experiment Simulation. *Aerospace Research in Bulgaria*, 22, pp. 59 – 67.
6. **Atanassov**, At., 2013a, An Adaptive Parallel Integrator of Ordinary Differential Equations System for Space Experiment, Simulation, *Proceedings of SES,* 2012, pp. 203 – 208.
7. **Atanassov**, At., 2013b, Program System for Space Missions Simulation - First Stages of Projecting and Realization. *Proceedings of SES,* 2012, pp. 209 – 214.
8. **Atanassov**, At., 2014a, Parallel Solving of Situational Problems for Space Mission Analysis and Design. *Proceedings of SES,* 2013, pp. 283 – 288.
9. **Atanassov**, A.M., 2014b, Parallel, adaptive, multi-object trajectory integrator for space simulation applications, *Advances in Space Research,* v. 54, 8, pp. 1581 – 1589.
10. **Atanassov**, At., 2015a, Method of Thread Management in a Multi-Pool of Threads Environments, *Proceedings of SES,* 2014, 12 – 14 November 2014, Sofia, Bulgaria, pp. 241 – 246.
11. **Atanassov**, At., 2015b, Development of Satellite Operation Scheduling Module for Space Mission Simulation Tool, *Proceedings of SES,* 2014, 12 – 14 November 2014, Sofia, Bulgaria, pp. 247 – 252.
12. **Atanassov**, A.M., 2016, Parallel satellite orbital situational problems solver for space missions design and control, *Advances in Space Research*, v. 58, 9, pp. 1819 – 1826.
13. **Atanassov**, A., 2017, Unification of „Pool of Threads” Control in the Frames of Different Parallel Solvers, *Proceedings of SES,* 2016, pp. 42 – 46.

Литература

1. Абалакин, В.К., Аксенов, Е.П., Гребеников, Е.А., Демин, В.Г., Рябов, Ю.А., 2012, Справочное руководство по небесной механике и астродинамике, – М., Наука, 864 с.
2. Авдюшев, В.А., 2010. Численное моделирование орбит. Томск: Изд-во НТЛ, 282 с.
3. Аксенов, Е.П., 1977. Теория движения искусственных спутников Земли. Наука. Гл. ред. физ.-мат. лит.
4. Atanasov, A.M., 1992, Methods for situational analysis based on the transformation of the situational condition towards the plane of Kepler's orbit, *Compt. rend. Acad. bulg. Sci*., vol. 45, no. 6, p. 49 – 52.
5. Atanassov, A., 2003. Analytical Effective Method for Verification of A Satellite Pass Over A Region of the Earth Surface. *Aerosp. Res. Bul*., 179 – 184.
6. Atanassov, A., 2009. An Analytical Method for Calculating the Satellite Bow Shock/Magnetopause Interception Positions and Times. *Compt. rend. Acad. bulg. Sci*., 62, No 1, pp. 97 – 104.
7. Бордовицына, Т.В., 1981, Метод Рунге-Кутта высоких порядков и стабилизирующие преобразования в задачах прогнозирования движения ИСЗ, *Космические исследования*, т.19. вып.6, с. 941-943.
8. Бордовицына, Т.В., 1984, Современные численные методы в задачах небесной механики. – М.: Наука. Главная редакция физико-математической литературы, 136 с.
9. Бордовицына, Т.В., Быкова, Л.Е., Кардаш, А.В., Федяев, Ю.А. and Шарковскй, Н.А., 1992. Эффективные алгоритмы численного моделирования движения ИСЗ. *Изв. вузов. Физика. Томск: Изд-во Том. ун-та*, *35*, pp.62-70.
10. Бордовицына, Т.В. and Авдюшев, В.А., 2016. Теория движения искусственных спутников Земли: аналитические и численные методы, Томск.
11. Гайдаров П., Суханов, А., Атанасов, Ат., Рязанова, В., 1989, Планирование астрофизических экспериментов в проекте "Шипка" Препринт Пр.-1534 , АН СССР.
12. Еременко, Р.П., 1965. Точное решение уравнения тени. Бюл Институт теоретической астрономии 6 (119), 446–449
13. Иванов, Д.С., Трофимов, С.П., Широбоков, М.Г., 2016, Численное моделирование орбитального и углового движения космических аппаратов. под общ. ред. д.ф.-м.н. М.Ю. Овчинникова. — М.: ИПМ им. М.В. Келдыша. —118 с.
14. Мещеряков, Г.А., Марченко, А.Н., 1982, О многоточечных моделях геопотенциала. В кн.: Изучение Земли как планеты методами астрономии, геодезии и геофизики. Киев, Наук. Думка, с. 121-131.
15. Назаренко, А.И., 2013. Моделирование космического мусора. *М.: ИКИ РАН*.
16. Прохоренко, В.И., 1976, Описание универсальной программы расчета навигационной информации о положении искусственного спутника Земли. Пр. ИКИ АН СССР № 263, с. 80.
17. Прохоренко, В.И., 1985, Ситуационный анализ орбит хвостового и аврорального зондов в проекте „ИНТЕРБОЛ". Пр. ИКИ АН СССР №1037, с. 70
18. Соколов, В.Г., 1980, О критериях пересечения орбиты спутника с тенью планеты. В „Наблюдения искусственных спутников Земли", БАН, сс. 152-157.
19. Советов Б.Я., Яковлев С.А., Моделирование систем, 3-е изд., перераб. и доп. — М.: Высш. шк., 2001. — 343 с.
20. Эльясберг, П.Е., 1965. Введение в теорию полета искусственных спутников Земли, М., „Наука", 1965, 540 с.
21. Эскобал, П., 1970. Методы определения орбит. *Пер. с англ*. М., „Мир".
22. Яшникова, О.М., 2011. Построение карт изолиний аномального гравитационного поля Земли на основе метода точечных масс. *Научно-технический вестник информационных технологий, механики и оптики*, 5, 75, сс. 35 – 39.
23. Aida, S., Kirschner, M., Wermuth, M., Kiehling, R. and Center, G.S.O., 2010. Collision avoidance operations for LEO satellites controlled by GSOC. *Space Operations: Exploration, Scientific Utilization, and Technology Development*, p.71.
24. Alowayyed, S., Groen, D., Coveney, P.V. and Hoekstra, A.G., 2017. Multiscale computing in the exascale era. *Journal of Computational Science*, *22*, pp.15-25.
25. Barbulescu, L., Howe, A.E., Whitley, L.D. and Roberts, M., 2004, June. Trading Places: How to Schedule More in a Multi-Resource Oversubscribed Scheduling Problem. In *ICAPS* (pp. 227-234).
26. Bartuschat, D. and Rüde, U., 2015. Parallel multiphysics simulations of charged particles in microfluidic flows. *Journal of Computational Science*, *8*, pp.1-19.

27. de Iaco Veris, A., 2018. Practical Astrodynamics. Springer International Publishing.
28. Cash, J.R. and Karp, A.H., 1990. A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software (TOMS)*, *16*(3), pp.201-222.
29. Dormand J. R., Prince P. J., 1987, Runge—Kutta— Nyström triples. *Comput. Math. Appl.*, 13, pp. 937-949.
30. Drake, J.B., Jones, P.W. and Carr Jr, G.R., 2005. Overview of the software design of the community climate system model. *The International Journal of High Performance Computing Applications*, *19*(3), pp.177-186.
31. Eickhoff, J., 2009, Simulating Spacecraft Systems. Springer-Verlag Berlin Heidelberg, 353.
32. Eickhoff, J., 2011. Onboard computers, onboard software and satellite operations: An introduction. Springer Science & Business Media.
33. Filippi, S., Gräf, J., 1986. New Runge–Kutta–Nyström formula-pairs of order 8 (7), 9 (8), 10 (9) and 11 (10) for differential equations of the form y ″= f (x, y). *Journal of computational and applied mathematics*, *14*(3), pp.361-370.
34. Gnoffo, P.A., 2007. A Perspective on Computational Aerothermodynamics at NASA. In *16th Australasian Fluid Mechanics Conference (AFMC)* (pp. 24-31). School of Engineering, The University of Queensland.
35. Gutiérrez, E., Asenjo, R., Plata, O. and Zapata, E.L., 2000. Automatic parallelization of irregular applications. *Parallel Computing*, *26*(13-14), pp.1709-1738.
36. Heath, M.T. and Jiao, X., 2004, June. Parallel simulation of multicomponent systems. In: International Conference on High Performance Computing for Computational Science (pp. 496-513). Springer, Berlin, Heidelberg.
37. Everhart E., 1974, Implicit Single Sequence Methods for Integrating Orbits, *Cel. Mech*., v. **10(1)**, pp. 35-55.
38. Fehlberg E., 1969, Klassische Runge-Kutta formeln funfter und siebenter Ordnung mit Schrittweitenkontrolle., *Computing*, v. **4**, p.93-106.
39. Fehlberg E., 1970, Klassische Runge-Kutta formeln vierter und nidrigerer Ordnung mit Schrittweitenkontrolle und ihre Anwendung auf Warmeleitungsprobleme., *Computing*, v. **6**, p. 61-71.
40. Fehlberg, E., 1972, Klassische Runge-Kutta-Nyström Formeln mit Schrittweiten-Kontrolle fur Differentialgleichungen X" = f(t, xl), Computing, **10(4)**, pp. 305-315.
41. Fehlberg, E., 1975, Klassische Runge-Kutta-Nyström-Formeln mit Schrittweiten-Kontrolle für Differentialgleichungen $\ddot{x} = f(t, x, \dot{x})$, *Computing*, **14(4)**, pp. 371-387.
42. Fortescue, P., Swinerd, G. and Stark, J. eds., 2011. *Spacecraft systems engineering*. John Wiley & Sons.
43. Gear, C.W., 1987. The potential for parallelism in ordinary differential equations. In: Fatunla, Simeon (Ed.), Computational Mathematics. Boole Press, Dublin, pp. 33–48.
44. GMAT, Available on URL: http://gmat.sourceforge.net/doc/R2015a/help.html
45. Gustafson, J.L., 1988. Reevaluating Amdahl's law. *Communications of the ACM*, *31*(5), pp.532-533.
46. Hairer, E., 1978. A Runge-Kutta method of order 10. *IMA Journal of Applied Mathematics*, *21*(1), pp.47-59.
47. Hairer, E., Nørsett, S.P. and Wanner, G., 1993. Solving ordinary differential equations. I, volume 8 of Springer Series in Computational Mathematics.
48. Hoots F.R., Crawford L.L. and Roehricg R.L., 1984, An Analytic Method to Determine Future Close Approaches Between Satellites, *Cel. Mech.* 33, 143-158.
49. Horn, M.K., 1983, Fourth and fifth-order scaled Runge-Kutta algorithms for treating dense output. SIAM *J.Numer. Anal*., Vol.20, p.558-568. *[II.6]*
50. Johnston, M.D., 1992, May. Spike: AI scheduling for Hubble Space Telescope after 18 months of orbital operations. In *Working Notes AAAI Spring Symposium on Practical Approaches to Scheduling and Planning*.
51. Jones, B.A., Born, G.H. and Beylkin, G., 2010, Comparisons of the cubed-sphere gravity model with the spherical harmonics. *Journal of guidance, control and dynamics*, *33*(2), pp.415-425.
52. Kholshevnikov, K. and Vassiliev, N., 1999, On the Distance Function Between Two Keplerian Elliptic Orbits. *Cel. Mech. Dyn. Astron* **75:** 75–83.
53. Kinnison, J.D., Maurer, R.H. and Jordan, T.M., 1990. Estimation of the charged particle environment for earth orbits. *Johns Hopkins APL Technical Digest*, *11*, pp.300-310.
54. Klinkrad, H., 2006. *Space debris: models and risk analysis*. Springer Science & Business Media.

55. Kramer, L.A. and Smith, S.F., 2006, June. Resource Contention Metrics for Oversubscribed Scheduling Problems. In: *ICAPS,* pp. 406-409.
56. Larson, J., Jacob, R. and Ong, E., 2005. The model coupling toolkit: a new Fortran90 toolkit for building multiphysics parallel coupled models. *The International Journal of High Performance Computing Applications*, *19*(3), pp.277-292.
57. Li, G., Zhu, Z.H., Ruel, S. and Meguid, S.A., 2017. Multiphysics elastodynamic finite element analysis of space debris deorbit stability and efficiency by electrodynamic tethers. *Acta Astronautica*, *137*, pp.320-333.
58. Miller, G., Lindenmayer, K., Johnston, M., Vick, S. and Sponsler, J., 1988. Knowledge based tools for Hubble Space Telescope planning and scheduling: constraints and strategies. *Telematics and Informatics*, *5*(3), pp.197-212.
59. Montenbruck, O., Gill, E., 2001, Satellite Orbits: Models, Methods, and Applications, Springer.
60. Nazirov, R., & Prokhorenko, V., 1996. The Multi-Satellite Space Missions and Campaigns Long-Term Planning. In *Space Mission Operations and Ground Data Systems-SpaceOps' 96*, p. 441.
61. Nazirov, R.R. and Prokhorenko, V.I., 1998. Situation analysis in the problems of space physics, *Kosm. Issl.*, vol. 36, no. 3, pp. 311–322.
62. Needham P.E. 1970, The formation and evaluation of detailed geopotential models based on point masses, Doctoral dissertation, The Ohio State University.
63. Neta, B., and Vallado, D., 1998. On Satellite Umbra/Penumbra Entry and Exit Positions, *Journal of the Astronautical Sciences*, 46, No. 1, 91–104.
64. Nyström, E.J., 1925, Über die numerische Integration von Differentialgleichungen, *Acta Soc. Sci. Fenn.* 50 (13).
65. Ortiz Longo, C.R. and Rickman, S.L., 1995. Method for the calculation of spacecraft umbra and penumbra shadow terminator points. NASA Technical Paper 3547
66. Papageorgiou, G. and Tsitouras, C., 1989. Scaled runge-kutta-nyström methods for the second order differential equation ÿ= f (x, y). *International journal of computer mathematics*, *28*(1-4), pp.139-150.
67. Pemberton, J.C., 2000. Towards scheduling over-constrained remote sensing satellites. In: *Proceedings of the 2d International Workshop on Planning and Scheduling for Space*.
68. Pemberton, J., Galiber F., 2001. A Constraint-Based Approach to Satellite Scheduling. In E. Freuder and R. Wallace, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science: Constraint Programming and Large Scale Discrete Optimization* 57:101-114.
69. Prince, P.J., Dormand, J.R., 1981, High-order embedded Runge-Kutta formulae, *J Comput. Appl. Math*., 7, pp. 67-76.
70. Prokhorenko, V.I., 1983a. STUDY OF SATELLITE SITUATIONS MISSION, *Acta Astronautica Vol,* 10, No. 7, pp. 499–503,
71. Prokhorenko, V.I., 1983b. Orbitaljnye tory v zadachah situatsionnyh issledovaniiy: Prepr. IKI AN SSSR 770, 24, Moscow,
72. Prokhorenko, V.I., 1985, SITUATION STUDIES OF AES ORBITS FOR SOLAR-EARTH SPACE EXPERIMENTS, *Acta Astronautica* Vol. 12, No 10. pp. 741–745.
73. Rabelo, L., Sala-Diakanda, S., Pastrana, J., Marin, M., Bhide, S., Joledo, O. and Bardina, J., 2013. Simulation modeling of space missions using the high level architecture. *Modelling and Simulation in Engineering*, *2013*, p.11.
74. Raghavachari, M. and Rogers, A., 1996. Understanding language support for irregular parallelism. In *Parallel Symbolic Languages and Systems* (pp. 174-189). Springer, Berlin, Heidelberg.
75. Rainey, L.B., 2004. Space Modeling and Simulation. The Aerospace Press, AIAA.
76. Rauber, T., Rünger G., 2010. Parallel Programming: For Multicore and Cluster Systems. Springer.
77. Rifai, S., Ferencz, R., Wang, W.P., Spyropoulos, E., Lawrence, C. and Melis, M., 1998, October. The role of multiphysics simulation in multidisciplinary analysis. In *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* (p. 4863).
78. Russell, R.P. and Arora, N., 2012. Global point mascon models for simple, accurate, and parallel geopotential computation. *Journal of Guidance, Control, and Dynamics*, *35*(5), pp.1568-1581.
79. Shafto, M., Conroy, M., Doyle, R., Glaessgen, E., Kemp, C., LeMoigne, J. and Wang, L., 2012. Modeling, simulation, information technology & processing roadmap. National Aeronautics and Space Administration.
80. Shampine, L.F., Gordon, M.K., Wisniewski, J.A., 1978, Variable order Runge-Kutta codes. In Computational Techniques for Ordinary Differential Equations Conference (Univ. of Manchester, 1978), I. Gladwell , D. K. Sayers, Eds. Academic Press, London, 83 – 101.

81. Shanks, E.B., 1965. Higher order approximations of runge-kutta type. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19650022581.pdf
82. Shanks, E.B., 1966. Solutions of differential equations by evaluations of functions. *Mathematics of Computation*, *20*(93), pp.21-38.
83. Simonsen, H.H., 1990. Extrapolation methods for ODE's: continuous approximations, a parallel approach (Ph. D. thesis). Mathematics and Science Division, Norwegian Institute of Technology, Trondheim, Norway.
84. Smith, S.F. and Pathak, D.K., 1992, March. Balancing antagonistic time and resource utilization constraints in over-subscribed scheduling problems. In *Artificial Intelligence for Applications, 1992., Proceedings of the Eighth Conference on*(pp. 113-119). IEEE.
85. Sommeijer, B.P., van der Houwen, P.J. and Neta, B., 1986. Symmetric linear multistep methods for second-order differential equations with periodic solutions. *Applied numerical mathematics*, *2*(1), pp.69-77.
86. Tóth, G., Sokolov, I.V., Gombosi, T.I., Chesney, D.R., Clauer, C.R., De Zeeuw, D.L., Hansen, K.C., Kane, K.J., Manchester, W.B., Oehmke, R.C. and Powell, K.G., 2005a, Space Weather Modeling Framework: A new tool for the space science community. *Journal of Geophysical Research: Space Physics*, *110*(A12).
87. Tóth, G., Volberg, O., Ridley, A.J., Gombosi, T.I., De Zeeuw, D.L., Hansen, K.C., Chesney, D.R., Stout, Q.F., Powell, K.G., Kane, K.J. and Oehmke, R.C., 2005b, A physics-based software framework for Sun-Earth connection modeling. In *Multiscale coupling of Sun-Earth processes* (pp. 383-397).
88. Tóth, G., Van der Holst, B., Sokolov, I.V., De Zeeuw, D.L., Gombosi, T.I., Fang, F., Manchester, W.B., Meng, X., Najib, D., Powell, K.G. and Stout, Q.F., 2012, Adaptive numerical algorithms in space weather modeling. *Journal of Computational Physics*, *231*(3), pp.870-903.
89. Tsitouras, C.H., Papakostas, S.N., 1999, Cheap Error Estimation for Runge Kutta methods*, SIAM J. Sci. Comput*., pp. 2067-2088.
90. Tsitouras, C., 2007. Runge–Kutta interpolants for high precision computations. *Numerical Algorithms*, *44*(3), pp.291-307.
91. Vallado, D.A., 2007. An analysis of state vector prediction accuracy. *Paper USR*, pp.07-S6.
92. Vallado, D.A., 2013 Fundamentals of Astrodynamics and Applications. Microcosm Press, third ed.
93. Vallado, D.A. and Finkleman, D., 2014. A critical assessment of satellite drag and atmospheric density modeling. *Acta Astronautica*, *95*, pp.141-165.
94. Vázquez, M., Houzeaux, G., Koric, S., Artigues, A., Aguado-Sierra, J., Arís, R., Mira, D., Calmet, H., Cucchietti, F., Owen, H. and Taha, A., 2016. Alya: Multiphysics engineering simulation toward exascale. *Journal of computational science*, *14*, pp.15-27.
95. Verner, J.H., 1978, *Explicit Runge-Kutta methods with estimates of the local truncation error*, SIAM J. Numer. Anal., pp. 772-790.
96. Wertz, J.R., Larson, W.J., 1999. Space Mission Analysis and Design. Microcosm Press, Kluwer Academic Publishers, third ed.
97. Wolfe, W.J. and Sorensen, S.E., 2000. Three scheduling algorithms applied to the earth observing systems domain. *Management Science*, *46*(1), pp.148-166.
98. Xian-He, S., Gustafson, J.L., 1991, Toward a better parallel performance metric. *Parallel Computing*, 17.10: 1093–1109.