

Міністерство освіти і науки України
Державний заклад
«Луганський національний університет імені Тараса Шевченка»

Навчально-науковий інститут математики та інформаційних технологій

Кафедра інформаційних технологій та систем

Стасюк Максим Вікторович

**РОЗРОБКА ПРОГРАМИ ДЛЯ ПРОЄКТУВАННЯ І РОЗРАХУНКУ
БУДІВЕЛЬНИХ КОНСТРУКЦІЙ**

кваліфікаційна робота

здобувача вищої освіти другого (магістерського) рівня

освітньої програми «Комп'ютерні мережі»

за спеціальністю 123 Комп'ютерна інженерія

Особистий підпис _____ Максим СТАСЮК

Науковий керівник _____ Володимир ДОНЧЕНКО,
старший викладач
кафедри інформаційних технологій
та систем

В.о. завідувача кафедри _____ Микола СЕМЕНОВ,
кандидат педагогічних наук, доцент
кафедри інформаційних технологій
та систем

Полтава – 2025

АНОТАЦІЯ

Стасюк М. В.

Тема: Розробка програми для проєктування і розрахунку будівельних конструкцій.

Спеціальність: 123 «Комп'ютерна інженерія».

Установа: ЛНУ імені Тараса Шевченка, 2025р.

Магістерська робота містить: 93 с., 24 рис., 1 табл., 50 джерел.

Об'єкт дослідження - будівельні конструкції і споруди.

Предмет дослідження - процес автоматизації проєктування плоских фермених конструкцій, який включає моделювання та аналіз напружено деформованого стану цих конструкцій.

Мета роботи - розробка програми для проєктування і розрахунку будівельних конструкцій.

Результати роботи – У роботі розглянуті основні завдання, які вирішуються в САПР, наведена узагальнена схема процесу автоматизованого проєктування. Розглянуто типову структуру САПР, яка базується на скінчено елементному аналізі, а також досліджено найпоширеніші в наш час вітчизняні та зарубіжні програмні комплекси моделювання та аналізу напружено деформованого стану будівельних конструкцій. В якості основного об'єкта розробки виступає фермова конструкція. Розглянуто евристичний підхід і підхід на основі методу силового аналізу. У роботі розглянуто методику проєктування силових схем фермених конструкцій методом силового аналізу. Розглядається програмна реалізація системи проєктування і розрахунку будівельних конструкцій.

В результаті розробки була отримана система проєктування і розрахунку будівельних конструкцій для проєктування фермених конструкцій.

Ключові слова: МЕТОД СКІНЧЕННИХ ЕЛЕМЕНТІВ, АРХІТЕКТУРА САПР, ФЕРМОВА КОНСТРУКЦІЯ, МЕТОД СИЛОВОГО АНАЛІЗУ.

ANNOTATION

Stasiuk Maksym

Theme: Development of a program for designing and calculating building structures.

Speciality: 123 " Computer Engineering ".

Institution: Luhansk Taras Shevchenko National University (LTSNU), 2025 year.

Master's work of: 93 p., 24 im, 50 sources.

Object of research is the building structures and structures.

Subject of research: is the process of automating the design of flat truss structures, which includes modeling and analysis of the stress-deformed state of these structures.

An aim of research is - development of a program for designing and calculating building structures.

The results of the work - The paper examines the main tasks that are solved in CAD, and presents a generalized scheme of the automated design process. A typical structure of CAD, which is based on finite element analysis, is considered, as well as the most common domestic and foreign software complexes of simulation and analysis of the stress-deformed state of building structures are studied. The main object of development is the truss structure. The heuristic approach and the approach based on the force analysis method are considered. The method of designing force schemes of truss structures by the force analysis method is considered in the work. The software implementation of the system of design and calculation of building structures is considered.

As a result of the development, a system of design and calculation of building structures for the design of truss structures was obtained.

Keywords: FINISHED ELEMENTS METHOD, CAD ARCHITECTURE, ROOF STRUCTURE, POWER ANALYSIS METHOD.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІМ	Інформаційна модель
МСЕ	Метод скінчених елементів
НДС	Напружено деформований стан
ООП	Об'єктно-орієнтоване програмування
ППП	Пакет прикладних програм
САПР	Система автоматизованого проектування
СЕ	Скінчений елемент
ТОК	Теоретично оптимальна конструкція

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	4
ВСТУП.....	7
РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ПРОЄКТУВАННЯ БУДІВЕЛЬНИХ КОНСТРУКЦІЙ І СПОРУД.....	10
1.1. Загальна схема процесу автоматизованого проєктування.....	10
1.2. Сучасні програмні засоби скінчено-елементного аналізу	13
1.3. Типова архітектура сучасних САПР	21
1.4. Загальна архітектура САПР, що базуються на методі скінчених елементів	25
1.5. Висновки до розділу	31
РОЗДІЛ 2. ПРОЄКТУВАННЯ ПЛОСКИХ ФЕРМЕННИХ КОНСТРУКЦІЙ	33
2.1. Проєктування силових конструкцій.....	33
2.2. Постановка задачі	36
2.3. Структурна оптимізація. Вибір силової схеми ферменної конструкції	41
2.3.1. Метод силового аналізу.....	42
2.3.2. Оцінка ефективності силових схем.....	44
2.3.3. Розрахунок силової ваги теоретично оптимальної конструкції	46
2.4. Розрахунок напружено-деформованого стану.....	47
2.4.1. Розрахунок напружено-деформованого стану ферменної конструкції.....	47
2.5. Висновки до розділу	49
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМИ ДЛЯ ПРОЄКТУВАННЯ І РОЗРАХУНКУ БУДІВЕЛЬНИХ КОНСТРУКЦІЙ	50
3.1. Обґрунтування вибору середовища розробки системи проєктування і розрахунку будівельних конструкцій	50
3.2. Загальна структура і функціональна схема системи проєктування і розрахунок будівельних конструкцій	54
3.3. Опис використання підсистеми "ферменна конструкція" в системі проєктування і розрахунку будівельних конструкцій.....	58

3.3.1. Загальні відомості	58
3.3.2. Початок роботи	60
3.3.3. Побудова ферменної конструкції.....	61
3.3.4. Розрахунок на міцність ферменної конструкції	76
3.4. Приклад розробки силової схеми ферменної конструкції.....	77
3.5. Висновки до розділу	83
ВИСНОВКИ	85
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	87
ДОДАТОК А.....	93

ВСТУП

Розвиток сучасної техніки, зокрема, створення високошвидкісних машин, енергетичних установок, аерокосмічної техніки і інших складних інженерних конструкцій і споруд, оцінка їх міцності і довговічності неможливі без наявності надійних і ефективних систем автоматизації проєктування (САПР). Рівень розвитку обчислювальної техніки сприяє постійному підвищенню її ролі: без використання комп'ютерів в даний час практично неможливо проєктування і створення надскладних технічних систем.

Автоматизація проєктування проявляється в наступних трьох основних аспектах: автоматизація рутинних інженерних робіт, автоматизація моделювання та аналізу властивостей проєктованого об'єкта і автоматизація завдань проєктування, що не підлягають повній формалізації.

Одним з найбільш важливих є другий аспект автоматизації проєктування, що полягає в заміні дорогого і тривалого експериментального дослідження дослідного зразка чисельним експериментом, суть якого полягає в побудові і дослідженні за допомогою комп'ютера математичної моделі проєктованого об'єкта. Крім того, на практиці не завжди є можливість випробувань дослідних зразків, так як це може привести до дуже серйозних економічних витрат, а іноді і до катастрофічних наслідків.

При проєктуванні складних будівельних конструкцій найбільш важливим елементом аналізу є дослідження їх напружено-деформованого стану, що призводить до необхідності автоматизації рішення за допомогою ЕОМ задач механіки деформованого твердого тіла.

Всі досягнення в області автоматизації проєктування інженерних конструкцій і споруд за допомогою комп'ютера, а також оцінка їх міцності в останні роки, були пов'язані з широким впровадженням та застосуванням комп'ютерних систем автоматизації проєктування; автоматизацією обчислювальних процесів і розробкою нових чисельних методів.

Найбільш доступним для аналізу силової роботи конструкцій є завдання, в яких реалізується плоский напружено-деформований стан. На плоских

конструкціях простіше розглядати раціональні способи і закономірності передачі зусиль. Ці конструкції досить легко сприймаються і досить просто зображуються графічно. Скінченно-елементні моделі (СЕМ) плоских конструкцій мають значно меншу розмірність в порівнянні з просторовими схемами, що спрощує не тільки їх побудову і скорочує час підготовки вихідних даних, але і полегшує обробку результатів розрахунків. Слід зазначити, що конструкції багатьох технічних об'єктів, наприклад, планера літака, є каркасними конструкціями з працюючими в мембранному режимі. Багато елементів каркаса: шпангоути, нервюри, стінки лонжеронів і т. д. - також знаходяться в плоскому напруженому стані. Тобто, плоскі задачі складають значну частину в загальному обсязі виконуваних реальних проєктів силових конструкцій.

Об'єкт дослідження - будівельні конструкції і споруди.

Предмет дослідження - процес автоматизації проєктування плоских фермених конструкцій, який включає моделювання та аналіз напружено деформованого стану цих конструкцій.

Мета роботи - розробка програми для проєктування і розрахунку будівельних конструкцій.

Методи дослідження: методи обчислювальної математики і комп'ютерної графіки.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- розглянути архітектуру САПР, що базуються на методі скінчених елементах;
- провести аналіз обчислювальних комплексів на основі МСЕ;
- розглянути проєктування силових конструкцій на прикладі форменої конструкції;
- розробити та реалізувати програми для проєктування і розрахунку будівельних конструкцій.

У першому розділі розглянуті основні завдання які вирішуються в САПР, наведена узагальнена схема процесу автоматизованого проєктування. Розглянута типова структура САПР, яка базується на звичайно елементному аналізі, а також досліджено найпоширеніші в наш час вітчизняні та зарубіжні програмні комплекси моделювання та аналізу напружено деформованого стану складних інженерних конструкцій і споруд.

У другому розділі розглянуто проєктування плоских фермених конструкцій. Розглянуто евристичний підхід і підхід на основі методу силового аналізу. В даному розділі розглянуто методику проєктування силових схем ферменних конструкцій методом силового аналізу і розрахунок напружено-деформованого стану ферменної конструкції.

Третій розділ присвячений розробці системи проєктування плоских конструкцій. Система реалізована на РС для роботи під управлінням ОС Microsoft® Windows 10 в системі програмування для Windows - Borland Delphi 7 і Visual Fortran. В даному розділі розглядається загальна структура і функціональна схема системи проєктування і розрахунку будівельних конструкцій і опис використання системи для проєктування плоских ферменних конструкцій.

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ПРОЄКТУВАННЯ БУДІВЕЛЬНИХ КОНСТРУКЦІЙ І СПОРУД

Створення сучасної техніки неможливо без наявності ефективних систем автоматизації проєктування, що дозволяють не тільки спростити і формалізувати процес опису і будівельних конструкцій, а й виконувати аналіз їх напружено-деформованого стану. Рівень розвитку обчислювальної техніки в даний час дозволяє ефективно застосовувати її для вирішення таких завдань. Всі досягнення в області автоматизації проєктування будівельних конструкцій за допомогою комп'ютера, а також оцінка їх міцності в останні роки, були пов'язані з широким впровадженням та застосуванням комп'ютерних систем автоматизації проєктування; автоматизацією обчислювальних процесів і розробкою нових чисельних методів.

1.1. Загальна схема процесу автоматизованого проєктування

Під проєктуванням розуміється процес створення опису, необхідного для побудови нового ще не існуючого об'єкта. Процес проєктування можна умовно поділити на два основних види діяльності: визначення технічних рішень, необхідних для розробки нового об'єкта, та створення комплексної документації, яка регламентує процес виготовлення цього об'єкта на виробництві і враховує обрані на першому етапі проєктні рішення.

Завдання проєктування можна розділити на синтез і аналіз. Під синтезом розуміється побудова формального опису об'єкта проєктування по заданому функціонуванню. Даний клас завдань пов'язаний зі створенням, як самого проєкту, так і проєктної документації.

Аналіз означає визначення того, як об'єкт працює відповідно до наданого опису. Завдання аналізу полягають у оцінці документів проєкту.

Загальну схему процесу автоматизованого проєктування можна представити таким чином (рис. 1.1) [16].

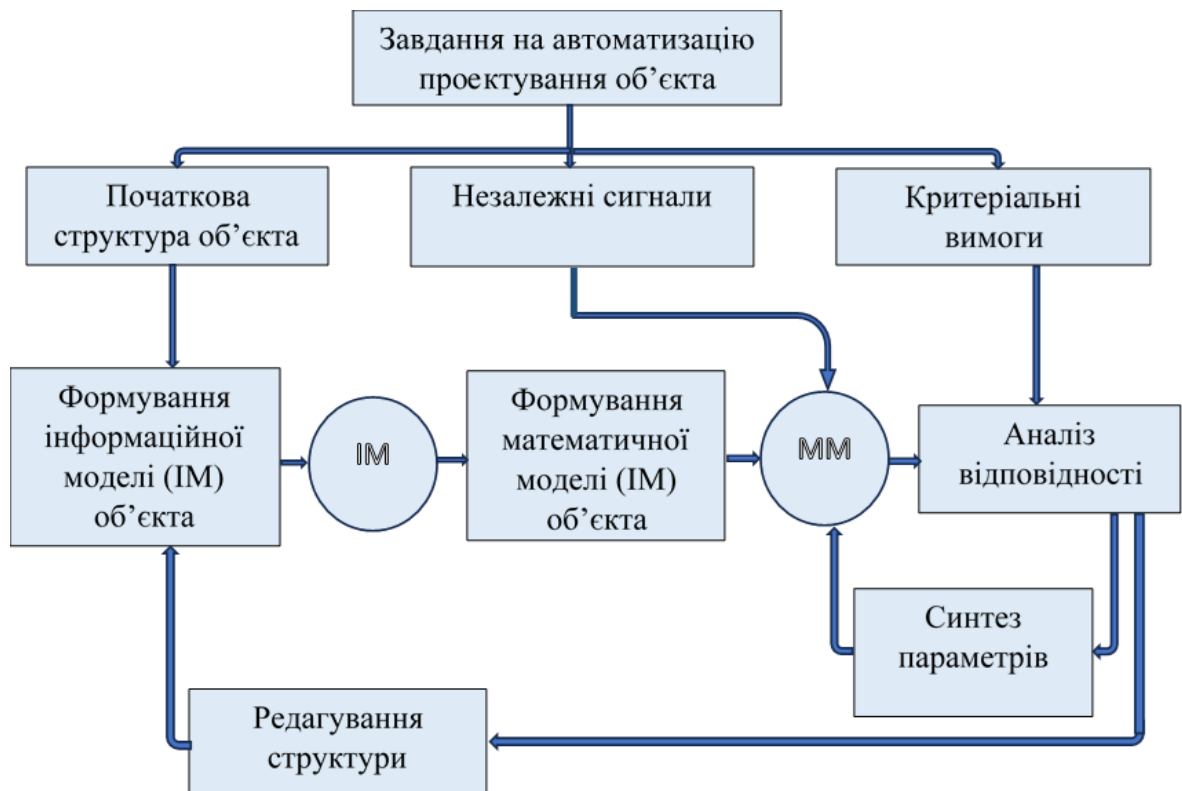


Рис. 1.1. Узагальнена схема процесу автоматизованого проектування

Формування структури проєктованого об'єкту, як правило, здійснюється шляхом композиції деяких базових структурних елементів, що зберігаються в бібліотеці опису моделей конкретної САПР. Опис структури виконується на деякій формальній мові, і представляється інформаційною моделлю.

Для аналізу об'єкта проєктування необхідна побудова математичної моделі, яка є центральним елементом структури будь-САПР. Математичну модель можна представити як сукупність математичних об'єктів (чисел, матриць, множин, функцій, операторів) і відносин між ними, яка адекватно відображає властивості проєктованого технічного об'єкта. У наведеній на рис. 1.1 схемою математична модель це деякий оператор, що зв'язує вхідні впливу і вихідні параметри. В теорії САПР в основному використовуються функціональні і структурні математичні моделі, детальна класифікація яких приведена в роботах [25,26].

Апаратом автоматизованого проєктування є засоби формування математичної моделі і синтезу конструкторських рішень. В силу того, що САПР спеціалізується по галузям застосування, наприклад, машинобудування

або проєктування радіоелектронної апаратури та обчислювальної техніки, вид математичної моделі безпосередньо залежить від типу об'єкту, що проєктується.

У машинобудуванні і будівництві застосовується велика кількість методів конструювання. Однак, всі САПР тут можна умовно поділити на дві великі групи: автоматизація виготовлення креслень проєктованого об'єкта, і проведення обчислень, пов'язаних з аналізом міцності та інших характеристик об'єкта, що проєктується. Їх дослідження можна звести до аналізу напружено-деформованого стану об'єкта проєктування, функціональною моделлю якого в цьому випадку буде система диференціальних або інтегральних рівнянь. Для їх вирішення сучасні САПР використовують різні методи лінеаризації [12,16].

Одним з найбільш ефективних чисельних методів, що дозволяють виконувати аналіз напружено-деформівного стану проєктованого об'єкта, є метод скінчених елементів (МСЕ). За його допомогою можна вирішувати завдання з області статички, динаміки, гідро- і аеродинаміки, теплопровідності, механіки суцільних середовищ, механіки ґрунтів.

Застосування МСЕ накладає певні вимоги на САПР. Для розрахунків з використанням даного чисельного методу проєктований об'єкт повинен бути представлений за допомогою скінченого числа елементарних обсягів, які називаються скінченими елементами (СЕ). Завдання дискретизації об'єкта на СЕ є досить складною і трудомісткою (особливо в тривимірному випадку). Це призводить до необхідності введення в САПР модуля, що дозволяє автоматизувати даний процес.

МСЕ заснований на фізичній ідеалізації, тобто одне з двох умов рівноваги (статичного або кінематичного) виконується точно, а інше наближено. Це призвело до того, що МСЕ застосовується в двох видах: у вигляді методу сил і методу переміщень [24]. На практиці частіше застосовується метод переміщень, так як при його використанні легко представляються кінематичні зв'язки для довільних конструкцій. Однак, в

деяких випадках більш доцільним є метод сил, що призводить до необхідності реалізації в САПР деякого механізму вибору необхідного методу розрахунку.

При застосуванні МСЕ виникає система лінійних алгебраїчних рівнянь (СЛАР), яка в загальному випадку може мати вельми високий порядок. Для її вирішення доводиться використовувати спеціальні модифікації методів вирішення систем рівнянь, як прямі, так і ітераційні. Після рішення СЛАР необхідно обчислювати додаткові параметри напружено-деформованого стану (наприклад, за отриманими переміщенням обчислювати деформації та напруги). Це призводить до появи великих масивів чисельних даних, які, в першу чергу, доводиться досліджувати на точність, достовірність і відповідність фізичним змістом завдання. Даний процес за часом і трудомісткістю може іноді перевершувати всі попередні етапи проєктування та розрахунків.

Таким чином, структуру САПР, що базуються на застосуванні МСЕ, можна умовно розділити на три взаємозв'язані групи:

- а) підготовка початкових даних для розрахунків, включаючи опис об'єкта конструювання, дискретизації його на КЕ, опис навантажень, фізичних і геометричних параметрів, рівняння стану та крайових умов;
- б) розрахунок з використанням методу скінченних елементів;
- в) аналіз отриманих результатів, включаючи синтез додаткових параметрів напружено-деформованого стану на основі раніше обчислених.

В даний час існує велика кількість САПР і обчислювальних комплексів, що використовують МСЕ в якості базового чисельного методу.

1.2. Сучасні програмні засоби скінчено-елементного аналізу

Розвиток методу скінчених елементів обумовлено взаємозв'язком трьох факторів: наявністю високопродуктивної обчислювальної техніки; розробкою математичних моделей досліджуваних явищ, адекватних реальним процесам з достатнім ступенем точності; особливостями самого методу [26].

У 60-х роках були створені перші програмні комплекси, в яких вперше був використаний метод скінчених елементів. Серед них були STRUDL-II, SAP-IV, NONSAP, ASKA, NASTRAN, SESAM-69 та інші [41]. Запровадження цих універсальних програмних систем стало можливим завдяки розвитку високопродуктивних електронно-обчислювальних машин, таких як IBM-370. Починаючи з кінця 70-х років, в СРСР було створено кілька десятків програмних комплексів для різних ЕОМ, в яких був реалізований метод скінчених елементів. До цього переліку входять МІРАЖ [13], МОРЕ [27], КАСКАД-2 [28], МІЦНІСТЬ-75 [17], МСЕ/20 [10], МАРС [11], ПАРСЕК [7], ЛІРА [14], СПРІНТ [34], FEA [4] та інші програми [33].

У США та інших країнах, розвиток методу скінчених елементів (МСЕ) і необхідність проведення міцнісних розрахунків сприяли дальшому удосконаленню і розширенню вже існуючих програмних комплексів, а також створенню нових. Було розроблено сотні програмних комплексів, які призначені для наближеного розв'язання різноманітних завдань, охоплюючи не лише механіку деформованого твердого тіла, але й такі області, як гідродинаміка, акустика, електротехніка і інші. Деякі з найбільш поширених серед них включають ABAQUS, ADINA, ASKA/DYNAN, ANSYS [2], MARC, MSC/NASTRAN [25], EUFEMI, COSMOS, HERCULE, MODULEF, SAP-7, LS-DYNA.

У кожної програми є свої переваги і недоліки при проведенні розрахунків для конкретної конструкції. Вибір конкретної програми для розрахунків залежить від рівня підготовки користувача в його науковій галузі, характеру задачі, типу доступної електронно-обчислювальної техніки (ЕОМ), розмірності задачі та інших факторів.

До критеріїв, що допомагає зробити вибір, слід віднести наступні фактори:

- програма широко використовується;
- в програмі використовуються новітні наукові досягнення;
- програма комерційно цілком доступна;

- є вичерпна і зрозуміла документація.

Вивчення програмної документації та наявної літератури, що описує програму та її складові, дозволяє зробити остаточний висновок про обґрунтованість вибору конкретного програмного комплексу.

Для МСЕ характерні особливості, які слід враховувати при виборі і розробці програми розрахунку. Такими особливостями є великі обсяги вихідних даних, проміжних і остаточних результатів розрахунку. Тому розрахунок по МСЕ складається з трьох основних етапів:

- розробка розрахункової кінцево-елементної схеми і підготовка вихідних даних;
- перевірка самого розрахунку;
- обробка результатів розрахунку.

Нижче на рисунку наводиться одна з можливих схем організації розрахунку за МСЕ [33]. Кожен етап є самостійною задачею. На початковому етапі найбільш важливим є створення початкової скінчено-елементної розрахункової моделі, виходячи з інженерної інтуїції про поведінку конструкції. Далі цю модель можна виправляти, враховуючи аналіз результатів розрахунку. Коригування моделі може виконуватися і за допомогою програмного забезпечення, якщо така можливість передбачена у програмному комплексі. Зазвичай, підготовка вихідних даних виконується за допомогою програм-генераторів сіток кінцевих елементів, що є частиною блоку підготовки даних.

Проведення розрахунку (етап 2) здійснюється розрахунковим блоком, в якому використовується той чи інший алгоритм розрахунку методом кінцевих елементів. Як правило, розрахунковий блок складається з ряду програмних модулів, кожен з яких виконується на певному етапі алгоритму. У найпростішому випадку програмної реалізації МСЕ для лінійної статичної крайової задачі теорії пружності розрахунковий блок містить наступну послідовність кроків:

- введення вихідних даних (наприклад, підготовлених програмою-генератором в окремому файлі);
- обчислення матриць жорсткостей скінчених елементів;
- формування глобальної матриці жорсткості повної структури;
- формування глобального вектора навантажень;
- рішення системи лінійних алгебраїчних рівнянь;
- обчислення переміщень вузлів сітки скінчених елементів, деформацій і напружень в довільних точках скінчених елементах.

На різних етапах розрахункового блоку включаються перевірки правильності вихідних даних і результатів проміжних обчислень (діагностика помилок), програмні модулі вибору поєднань навантажень, що діють на конструкцію, визначення площі перетинів арматури в залізобетонних конструкціях і інші. Діагностика помилок на етапі виконання програми є важливою, так як при своєчасному виявленні помилки припиняються обчислення, що призводить до економного використання ресурсів ЕОМ.

Схема організації розрахунку за МСЕ



Проведення розрахунку (етап 2) здійснюється розрахунковим блоком, в якому використовується той чи інший алгоритм розрахунку методом кінцевих елементів. Як правило, розрахунковий блок складається з ряду програмних модулів, кожен з яких виконується на певному етапі алгоритму. У найпростішому випадку програмної реалізації МСЕ для лінійної статичної крайової задачі теорії пружності розрахунковий блок містить наступну послідовність кроків:

- введення вихідних даних (наприклад, підготовлених програмою-генератором в окремому файлі);
- обчислення матриць жорсткостей скінчених елементів;
- формування глобальної матриці жорсткості повної структури;
- формування глобального вектора навантажень;
- рішення системи лінійних алгебраїчних рівнянь;

- обчислення переміщень вузлів сітки скінчених елементів, деформацій і напружень в довільних точках скінчених елементів.

На різних етапах розрахункового блоку включаються перевірки правильності вихідних даних і результатів проміжних обчислень (діагностика помилок), програмні модулі вибору поєднань навантажень, що діють на конструкцію, визначення площі перетинів арматури в залізобетонних конструкціях і інші. Діагностика помилок на етапі виконання програми є важливою, так як при своєчасному виявленні помилки припиняються обчислення, що призводить до економного використання ресурсів ЕОМ.

Ефективне використання електронно-обчислювальних машин також досягається через розробку спеціальних методів вирішення стандартних математичних задач, які враховують особливості методу скінчених елементів (МСЕ), зокрема стрічковість і розрідженість матриці жорсткості розрахункової моделі конструкції. При обчисленні напружено-деформованого стану конструкції за лінійною теорією пружності при впливі статичних навантажень, виникає задача вирішення системи лінійних алгебраїчних рівнянь. У програмних комплексах для скінчено-елементного аналізу використовуються різноманітні методи для вирішення великих систем лінійних рівнянь.

Різні варіанти методу Гаусса реалізовані в програмах ADINA (блоковий метод Гаусса), ASKS, SAP-7 (стрічковий метод Гаусса), NASTRAN (LTDL - декомпозиція). Ефективним є фронтальний метод, реалізований в програмах ABAQUS, ANSYS та ін. Методи суперелементів і редукції базису дозволяють істотно скоротити час обчислення [18]. Ефективними є також ітераційні методи.

Розрахунок власних коливань конструкції виконується методами: ітерації в підпросторі (SAP-7), обчислення коренів характеристичного визначника (NASTRAN), Хаусхолдера з використанням методу Якобі (ASKA), Гивенса і QR-методу (NASTRAN), Hiber-Hughes (ABAQUS), Ланцоша (PKM). При розрахунок динамічного відгуку використовуються методи: подання

рішення у вигляді суперпозиції форм власних коливань, крокові - Вілсона, Ньюмарка (ABAQUS, ADINA, SAP-7, NASTRAN). Рішення геометрично і фізично нелінійних задач здійснюється, як правило, ітераційними методами, основу яких складає метод Ньютона-Рафсона в поєднанні з кроковими методами (ABAQUS, ADINA, NASTRAN, ANSYS, LSTRAN і ін.) [33].

Слід зазначити, що принцип модульності програмування, використаний в програмних комплексах, дозволяє створювати як універсальні обчислювальні програми, так і промислові для вирішення вузького класу задач. На перших етапах освоєння МКЕ розроблялися в основному промислові обчислювальні програми. Вони ефективні, якщо вирішується велика кількість варіантів однотипних завдань, або виконується великий обсяг обчислень для якісного і кількісного дослідження явищ, пов'язаних з новою постановкою завдання. Тенденція розвитку обчислювальної техніки, яка призвела до створення персональних ЕОМ і нових інформаційних технологій, вплинула на розробку програмного забезпечення МСЕ [6]. Програмні комплекси по МСЕ активно використовуються в системах автоматизованого проектування, які базуються на персональних ЕОМ (AutoCAD / MechanicalDesktop, Pro-Engineer).

Сучасні комплекси програм, в яких використовується МСЕ, дозволяють отримувати наближені чисельні рішення при розрахунку конструкцій на статичні і динамічні навантаження для широкого класу матеріалів з різними механічними характеристиками і поведінкою. Розрахунок конструкцій на статичні навантаження може проводитися з урахуванням фізичної і геометричної нелінійності, температурних полів, взаємодії з іншими середовищами (наприклад, з рідиною). Проводиться розрахунок критичних навантажень, при яких конструкція або її елементи втрачають стійкість, поведінки конструкції після втрати стійкості. МСЕ дозволяє також визначити навантаження, при яких відбувається руйнування конструкції. Враховуються такі властивості матеріалу як анізотропність, нелінійна пружність, пластичність, текучість. Враховуються види геометричної нелінійності: великі

деформації і великі переміщення. Основними динамічними завданнями є: розрахунок власних коливань конструкції; динамічний відгук на навантаження, що залежить від часу поширення хвиль [33].

У таблиці 1.1. наводяться порівняльні характеристики найбільш поширених комплексів програм описаних в роботі [40].

Наведені в таблиці комплекси програм є універсальними. Слід зазначити, що вибір моделі поведінки матеріалу є визначальним для достовірності розрахованого напружено-деформованого стану конструкції. Наприклад, в програмі MARS використовується 30 моделей поведінки матеріалу (в'язкопружних по Максвеллові або Кельвіном; пластичність за критерієм Мізеса або Мора, або Кулона, з ізотропним або (і) кінематичним зміцненням по теорії асоційованого або неасоційованих течії, з урахуванням температури; повзучість за законом Мізеса, чисто об'ємна або чисто девіаторна повзучість, анізотропність при пружно-пластичній повзучості: в'язкопластичні, нестисливі або майже нестисливі, великі деформації за законом Муні-Рівлін і т.д.).

Таблиця 1.1

Порівняльні характеристики найбільш поширених комплексів програм

	ABAQUS	ADINA	ANSYS	ASKA	COSMOS	MARS	MODULEF	NASTRAN	PAFEC	SAP-7	SESAM-80	TITUS
Статистичні навантаження	+	+	+	+	+	+	++	+	+	+	+	+
Динамічні навантаження	+	+	+	+	+	+		+	+	+	+	+
Облік геометричної нелінійності	+	+	+	-	+	+	+	+	+	+	-	+
Стійкість	+	+	+	+	+	+	-	+	+	+	-	+
Розрахунок руйнувань	+	+	+	+	+	+	-	-	+	+	+	+
Облік контакту	+	+	-	+	+	+	-	+	-	+	-	-
Облік температурного поля	+	+	+	+	+	+	+	+	+	+	-	+

	ABAQUS	ADINA	ANSYS	ASKA	COSMOS	MARS	MODULEF	NASTRAN	PAFEC	SAP-7	SESAM-80	TITUS
Взаємодія з рідиною	-	+	+	-	+	-	-	+	+	-	+	+
ВЛАСТИВОСТІ МАТЕРІАЛУ:												
Ізотропність	+	+	+	+	+	+	+	+	+	+	+	+
Ортоотропність	+	+	+	+	+	+	+	+	+	+	-	+
Анізотропність	+	+	+	+	+	+	+	+	+	+	-	+
Пружно-пластичність	+	+	+	+	+	+	+	+	+	+	+	+
Нелінійна пружність	+	+	+	+	+	+	+	+	+	+	-	+
В'язкопружність з плавучістю	-	+	+	+	+	+	-	+	+	-	-	+
Грунт	+	+	-	+	-	+	-	-	+	-	+	+
бетон	-	+	+	-	-	+	-	-	+	-	-	-

Програми промислового призначення призначені для розрахунку вузького класу конструкцій. Прикладом програми промислового призначення є програма BERSAFE, розроблена для розрахунку елементів конструкцій атомної енергетики. У цій програмній системі для розрахунку напружено-деформованого стану використовуються спеціальні закони повзучості бетону і графіту, поведінки скельної породи. Іншим прикладом програми промислового призначення служить програма EFESYS, призначена для розрахунку гребель і морських споруд з урахуванням пов'язаності процесів фільтрації і напружено-деформованого стану. Для вивчення різних аспектів МКЕ розроблені спеціальні навчальні програмні комплекси [27]. Для вирішення різних завдань біомеханіки широко використовуються комерційні універсальні комплекси ABAQUS, ANSYS, MARS, NASTRAN, а також розробляються спеціалізовані програми, що реалізують метод кінцевих елементів, для підвищення ефективності вирішення тієї чи іншої конкретної задачі [29].

1.3. Типова архітектура сучасних САПР

Архітектурою називається сукупність і структура компонентів програми або системи, взаємозв'язку між ними, принципи і керівництва для їх проєктування і розвитку в часі. Сучасні системи автоматизації проєктування

мають типову архітектуру, що включає в себе такі елементи, як лінгвістичне, інформаційне та програмне забезпечення (рис. 1.2).

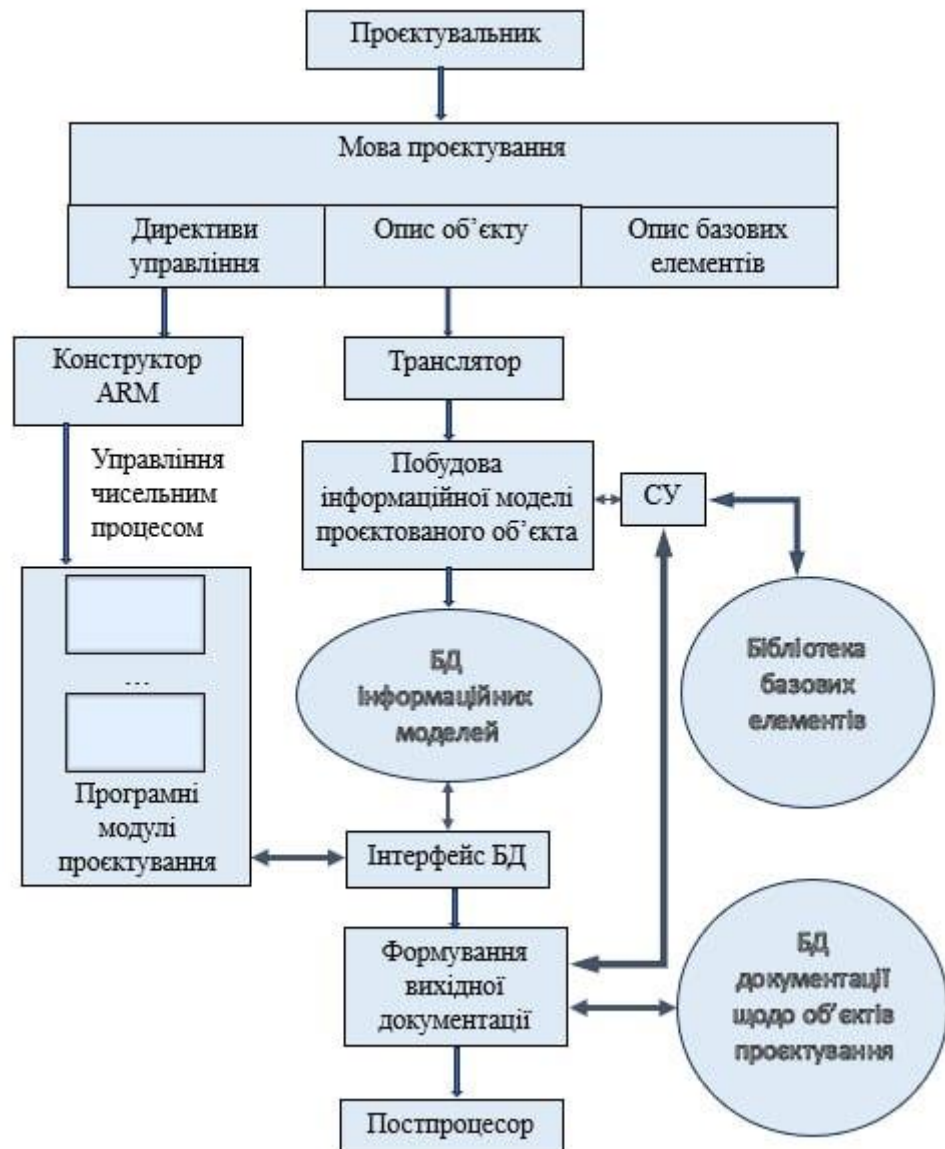


Рис. 1.2. Типова архітектура САПР

Лінгвістичне забезпечення САПР поділяється на три групи:

- а) мови користувача, призначені для його взаємодії з системою і для опису об'єкта проектування;
- б) мови внутрішнього представлення даних, призначені для опису інформаційної моделі об'єкта;
- в) мови машинних архівів, призначені для опису форматів зберігання спроектованого об'єкта у вигляді сукупності графічної і текстової інформації;

г) інструментальні мови, призначені для побудови проблемно-орієнтованого програмного забезпечення.

Інформаційне забезпечення САПР включає в себе три види баз даних (БД), що утворюють єдину БД САПР:

а) бібліотека базових елементів, що включає в себе елементи різного рівня і призначення, такі як:

- опис моделей структурних елементів, з яких будується об'єкт, що проєктується;

- опис форматів документації;

- нормативно-довідкова документація, необхідна для проєктування об'єкта (наприклад, ГОСТи і БНіП);

- опис допоміжних технологічних даних;

б) тимчасова допоміжна БД, призначена для зберігання інформаційної моделі об'єкта на різних етапах його проєктування;

в) БД, призначена для зберігання повного пакету документації по спроектованому об'єкту.

Інформаційна модель (ІМ) проєктованого об'єкта є ядром усього процесу проєктування конкретної конструкції. До складу ІМ входить структура об'єкта, описана на деякому вхідному мові і, як правило, складена з деякого набору базових об'єктів, що зберігаються у відповідній бібліотеці. Крім того, до складу ІМ входить математична модель (ММ), що описує цікаві для інженера-проєктувальника характеристики об'єкта або його стан.

Програмне забезпечення САПР включає в себе такі базові елементи, як препроцесор, процесор і постпроцесор, які зазвичай складаються з наступних програмних компонент:

а) АРМ проєктувальника;

б) транслятор вхідної мови;

в) модуль побудови ІМ проєктованого об'єкта;

г) СУБД САПР;

д) модулі проєктування, реалізують різні аспекти розрахунку створюваного об'єкта (наприклад, аналіз його напружено-деформованого стану);

е) інтерфейс з БД;

ж) підсистему формування проєктної (вихідний) документації.

АРМ проєктувальника - центральний орган управління всією САПР. Він забезпечує управління послідовністю виконання всіх проєктних завдань і діагностику можливих помилок.

Препроцесор - це пакет прикладних програм (ППП), призначений для автоматизації підготовки початкових даних, необхідних для проєктування конкретного об'єкта. Як правило, основна задача препроцесора полягає в автоматизації опису геометричної моделі проєктованої конструкції і дискретизації її на кінцеві елементи (КЕ). Крім того, за допомогою препроцесора зазвичай задають граничні і початкові умови, фізичні властивості конструктивних матеріалів, параметри розрахунку. До складу препроцесора як правило, входить транслятор вхідних мов завдань на виконання проєктних робіт. В типову структуру препроцесора також входить модуль побудови ІМ проєктованого об'єкта, призначений для формування повного набору вихідних даних, що утворюють ІМ об'єкта і необхідних для проведення подальших чисельних розрахунків.

Процесор - головна складова будь-якої САПР. Основне призначення процесора - виконання необхідних чисельних розрахунків при проєктуванні конкретної конструкції по заздалегідь побудованій ІМ. До складу процесора входять проблемно-орієнтовані програмні модулі, що реалізують необхідні в конкретній САПР види розрахунків. Кожен модуль виконує певну закінчену проєктну процедуру моделювання і синтезу проєктних рішень або аналізу сконструйованого об'єкта.

Постпроцесор призначений для автоматизації, як аналізу отриманих результатів розрахунку, так і випуску необхідної вихідної проєктної документації на конструюються об'єкт.

Питання організації архітектури складних програмних систем, до яких відносяться і САПР, є важливими напрямками теоретичних досліджень. Вони показують, що архітектура сучасного ПО будується за двома основними напрямками:

а) модульність, коли ПО має відкриту архітектуру, яка утворена поповнюється сукупністю програмних модулів, кожен з яких може при необхідності бути заміненим або використаним іншою програмою;

б) розвиток сучасних методів структурування архітектури ПО з використанням технологій клієнт-сервер або конвеєра.

Більшість сучасних САПР, таких, наприклад, як ANSYS, COSMOS, NASTRAN мають модульну структуру.

1.4. Загальна архітектура САПР, що базуються на методі скінчених елементів

Загальна структура

Практичний розрахунок характеристик деякого пристрою в процесі проєктування проходить стадію уявлення завдання рівняннями в приватних похідних і включає три етапи:

- опис геометрії, фізичних характеристик, генерацію мережі скінчених елементів;
- розрахунок за допомогою методу скінчених елементів;
- візуалізацію і інтерпретацію результатів моделювання.

Схема операцій при розв'язуванні
задачі методом скінчених
елементів



Ці три етапи добре розділені і в дійсності відповідають на рівні програмного забезпечення трьох функцій, виконуваних окремими модулями:

- модулем введення даних (препроцесором);
- модулем обчислень (процесором);
- модулем виводу результатів (постпроцесор).

Функції модуля введення

Модуль введення призначений для введення і підготовки всієї інформації, необхідної для вирішення завдання методом кінцевих елементів. Слід повідомити дані про дискретизації області і представити її фізичні характеристики. Модуль введення повинен також здійснювати наступні три функції:

- опис геометрії об'єкта;
- генерацію мережі кінцевих елементів;
- вказівка областей і меж.

Генерація мережі скінчених елементів в області полягає в формуванні сукупності вузлів і сукупності скінчених елементів, що забезпечують прийнятну дискретизацію області. Така дискретизація повинна відповідати кордонам області і внутрішнім кордонів між різними її ділянками. Крім того, кінцеві елементи не повинні мати форму, занадто відрізняється від симетричних форм стандартних елементів (рівносторонніх трикутників або тетраедрів, квадратів або кубів).

Вузли визначаються їх координатами, тоді як елементи характеризуються їх типом і переліком їх вузлів. Деякі формулювання завдань вимагають використання інтегралів на кордонах. У цьому випадку додатково до скінчених елементів області (об'ємним в тривимірних задачах, лінійним в двовимірних) потрібно створити кінцеві елементи кордонів (поверхневі в тривимірних і лінійні в двовимірних задачах, дискретізується розглядаються кордону).

Операція вказівки областей і меж дозволяє уточнити фізична поведінка:

- опис фізичних характеристик матеріалів (наприклад, провідність, теплопровідність і т. Д.);
- опис джерел (наприклад, джерела тепла);
- опис граничних умов;
- опис початкових умов для час змінних завдань.

Зазвичай ця інформація вводиться послідовно ділянка за ділянкою, межа за межею. Зв'язки між ділянками, кінцевими елементами області та вузлами дозволяють відобразити цю інформацію у вигляді дискретизації області.

Опис геометрії іноді проводиться в неявній формі при створенні мережі КЕ. Однак в даний час прагнуть розділити ці операції. Спочатку складається опис геометрії, а потім створюється мережа СЕ, яка використовує задану геометрію. Крайнім випадком є використання двох спеціалізованих програм: жорсткого моделювання для геометричної частини, і складання мережі СЕ для дискретизації.

Функції модуля обчислень

Модуль обчислень вирішує одиночне рівняння або систему лінійних або нелінійних рівнянь.



Цей модуль отримує на вході опис мережі, фізичні характеристики і граничні умови. На виході він видає значення шуканих величин в кожному вузлі мережі.

Для вирішення систем рівнянь використовуються два сімейства методів: методи точкові або блокові, що діють шляхом релаксації, і глобальні матричні методи. Останні методи, використовувані значно частіше, мають кілька етапів:

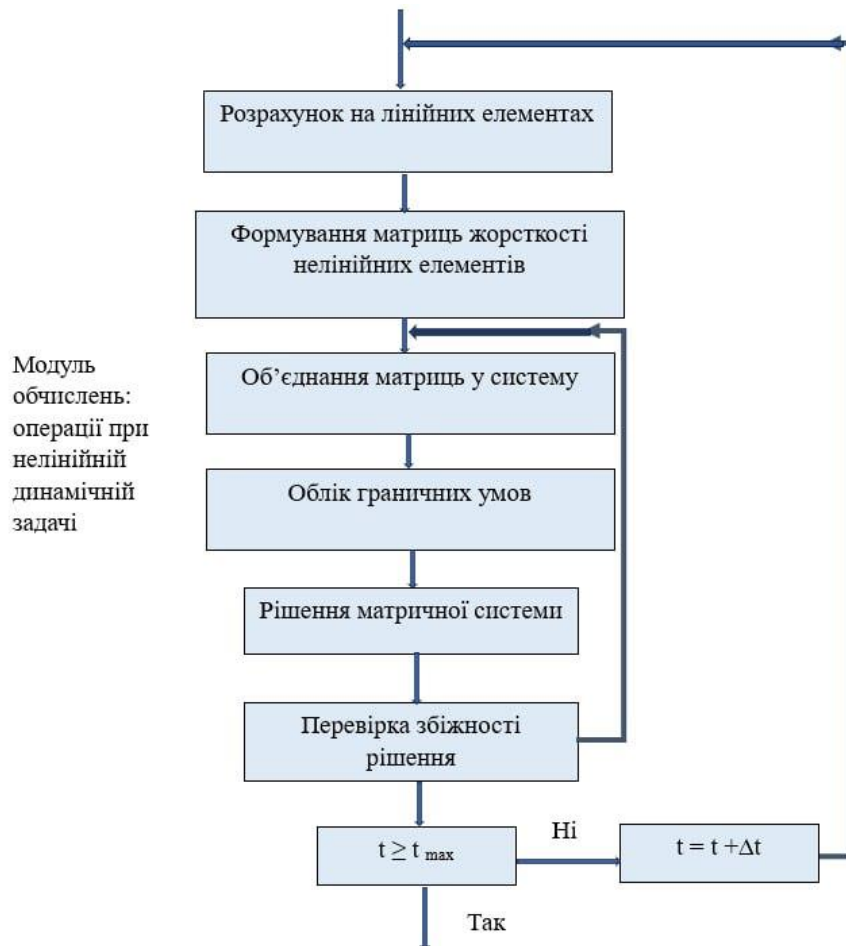
- побудова підматриць і власних підвектора на кожному кінцевому елементі;
- об'єднання цих підматриць і підвектора для формування матриці і правої частини;
- облік граничних умов;
- рішення лінійної системи.



Рішення лінійних систем здійснюється кількома можливими способами: прямими методами (Гаусса, Холецкого); променями методами; ітераційним блоковими методами (Гаусса-Зейделя).

Для систем нелінійних рівнянь ці операції повторюються відповідно до прийнятої ітераційної схеми (Гаусса-Зейделя, Ньютона-Канторовича, Ньютона - Рафсона).

Для змінних з часом завдань такий розгляд має бути повторений на кожному часовому кроці (явні і неявні методи кінцевих різниць Кренк-Ніколсона, прогнозу-корекції).



Функції модуля виведення

Модуль введення дозволяє описати завдання, які потім вирішуються модулем обчислень. Однак отримане рішення не може безпосередньо використовуватися з наступних причин:

- значення змінних в вузлах скінчено-елементній мережі не завжди мають чіткий фізичний зміст (наприклад, вектор магнітного потенціалу в задачах електромагнетизму);
- маса необробленої чисельної інформації, одержуваної при обчисленні (кілька тисяч вузових величин), занадто велика для сприйняття користувачем.

Модуль виведення грає подвійну роль:

- витягує значущу інформацію. Ця інформація може бути пов'язана з локальними величинами (наприклад, магнітної індукції, питомими втратами, механічними напруженнями і т. д.) або глобальними величинами (тепловим потоком, електромагнітними силами і т. д.);

- представляє чисельну інформацію в графічній формі для полегшення її сприйняття і інтерпретації (у вигляді карти полів, ізотерм, постійних механічних напружень, кривих зміни температури або магнітного поля уздовж деякої лінії і т. д.).

Загальна блок-схема розрахунку конструкцій методом кінцевих елементів

Наведемо як приклад загальну блок-схему розрахунку конструкцій за методом кінцевих елементів для задач міцності рис.1.3.



Рис. 1.3. Блок-схема розрахунку конструкцій за методом скінчених елементів для задач міцності

1. Завдання вихідної інформації

- розташування вузлових точок в загальній системі координат (координати вузлів);
- взаємне розташування кінцевих елементів (топология конструкції);
- значення геометричних і характеристик жорсткості параметрів кожного з елементів конструкції;
- значення зовнішніх вузлових, поверхневих і об'ємних сил.

2. Визначення вузлових точок елементів в місцевій системі координат.

3. Побудова матриці жорсткості для e -го елемента в місцевій системі координат $[k]^{(e)}$

4. Знаходження напрямних косинусів для кожного елемента і матриці перетворення переміщень з місцевою системи координат в загальну $[T]^{(e)}$.

5. Визначення матриці жорсткості елемента в загальній системі координат $[\bar{k}]^{(e)} = [T]_{(e)}^T [k]^{(e)} [T]^{(e)}$

6. Визначення загальної матриці жорсткості для всієї конструкції $[\bar{k}]$.

7. Накладення на конструкцію певного числа зв'язків, що виключають її переміщення як абсолютно жорсткого тіла. Останнє призводить до отримання деякої урізаною загальної матриці жорсткості $[\bar{k}^*]$.

8. Приведення поверхневих і об'ємних сил, а також початкових деформацій до еквівалентним вузловим силам $\{P_v\}_S$; $\{P_v\}_S$; $\{P_\varepsilon\}_S$;

9. Знаходження сумарних вузлових сил в кожному S-му вузлі

$$\{\bar{P}_*\}_S = \{P\}_S + \{P_v\}_S + \{P_v\}_S + \{P_\varepsilon\}_S$$

10. Відшукування матриці $[\bar{k}_*]^{-1}$

11. Визначення вузлових переміщень конструкції

$$\{\bar{q}\} = [\bar{k}_*]^{-1} \{\bar{P}_*\}_S$$

12. Визначення вузлових переміщень і компонентів напруги для кожного e -го кінцевого елемента

1.5. Висновки до розділу

У цьому розділі розглянуто основні завдання які вирішуються в САПР, наведена узагальнена схема процесу автоматизованого проєктування. Розглянута типова структура САПР, яка базується на звичайному елементному аналізі, а також досліджено найпоширеніші в наш час вітчизняні та зарубіжні програмні комплекси моделювання та аналізу напружено деформованого стану складних інженерних конструкцій і споруд.

Відзначено, що сучасні програмні комплекси автоматизації проєктування можна умовно розділити на три підсистеми: препроцесор,

процесор і постпроцесор. Препроцесор відповідає за підготовку вхідних даних, яка включає такі етапи роботи, як опис геометричної моделі проєктованого об'єкта, і його дискретизацію на заданий тип кінцевих елементів. Процесор виконує всі необхідні для конкретного типу завдання розрахунки: формує матриці мас, жорсткостей; будує систему лінійних алгебраїчних рівнянь; враховує початкові і граничні умови; вирішує систему рівнянь і виводить результати розрахунку. Постпроцесор автоматизує процес аналізу результатів і генерацію документації.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПЛОСКИХ ФЕРМЕНИХ КОНСТРУКЦІЙ

2.1. Проєктування силових конструкцій

Завдання проєктування конструкцій технічних об'єктів відносяться до числа найбільш поширених прикладних задач автоматизованого проєктування. Проєктовані конструкції повинні задовольняти не тільки вимогам міцності, жорсткості і стійкості, а й вимогам мінімальної вартості, матеріаломісткості та спеціальних вимог свого функціонального призначення. Вимоги до проєктованих конструкцій, поряд з обмеженнями функціонального характеру, нерідко визначаються необхідністю зниження маси одержуваних виробів.

При роботі споруд і машин їх частини сприймають зовнішні навантаження і дії цих навантажень передають один одному.

Тому при проєктуванні таких споруд інженеру доводиться вибирати матеріал і поперечні розміри кожного елемента конструкції так, щоб він міг працювати цілком надійно, без ризику зруйнуватися, або спотворивши свою форму, пручався дії зовнішніх сил, що передаються на нього від сусідніх частин конструкції. Конструкції, що сприймають зовнішні сили і передають ці сили на інші елементи споруди, інші конструкції або опори будемо надалі називати силовими конструкціями.

У проєктуванні так звані силові конструкції, тобто конструкції, що сприймають і передають навантаження, займають особливе місце. Якщо для визначення мас допоміжних і несилових конструктивних елементів статистика є, мабуть, єдиним засобом, особливо на початкових етапах проєктування, то для силових конструкцій статистика має лише допоміжне значення і для них необхідне застосування точних і складних методів розрахунку.

Проєктування силових конструкцій зазвичай проводиться в два етапи.

На *першому* етапі відбувається проєктування структури або структурна оптимізація. На цьому етапі відшукується силова схема або несуча структура конструкції, яка зумовлює генеральні шляхи передачі зусиль. Проєктування силових схем проводиться на основі аналізу характеру навантаження і

можливих умов закріплення. В кінцевому підсумку визначаються розташування і взаємозв'язок основних силових елементів конструкції.

На *другому* етапі проводиться так звана параметрична оптимізація, в процесі якої відбувається відшукування найкращих параметрів елементів конструкції, розробленої на попередньому етапі, а саме: значення товщини елементів, форм і площ поперечних перерізів і т. д. Але таке визначення завжди пов'язано з проведенням розрахунків на міцність і стійкість. Тому процес визначення найкращих параметрів проводиться ітеративно, тобто шляхом повернень і повторів і є частиною загального оптимізаційного процесу. Задачі параметричної оптимізації в достатній мірі формалізовані і успішно вирішуються з використанням різних математичних методів.

Значно менш формалізовані завдання структурної оптимізації. Безумовно, ні перший, ні другий етапи неможливі без розрахунку на міцність і стійкість проміжних і остаточних проєктних рішень. Тому для забезпечення якості та достовірності отриманих результатів найважливішу роль відіграють використовувані методи розрахунку, яким доводиться приділяти підвищену увагу особливо на пізніх стадіях розробки проєкту.

Традиційна схема проєктування силових конструкцій повністю відповідає наведеній вище схемі процесу проєктування (рис. 2.1).

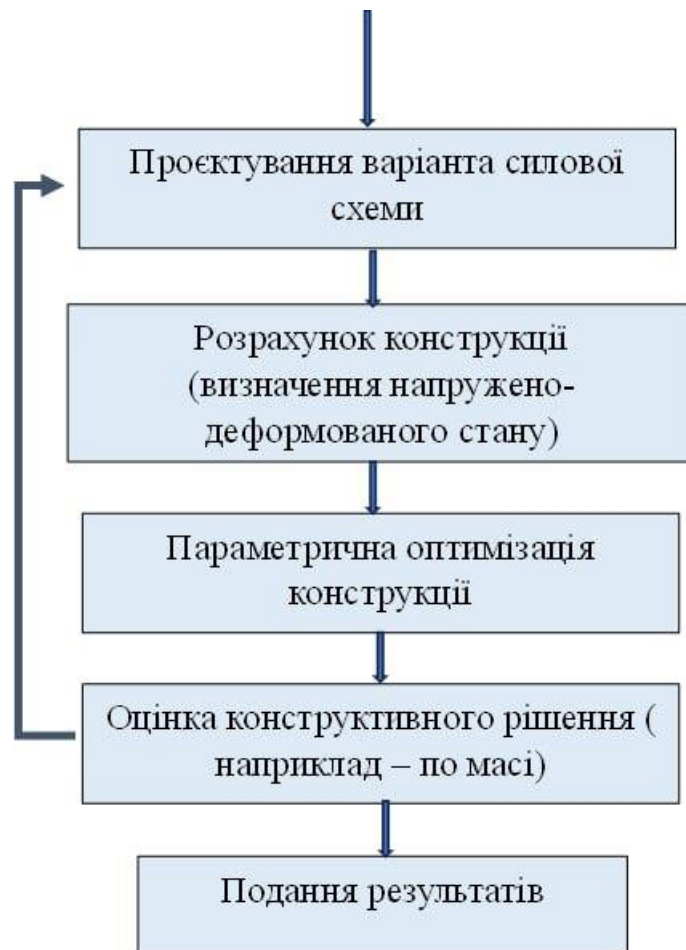


Рис. 2.1. Стандартна схема процесу проектування силових конструкцій

Силовая структура визначає загальний розподіл зусиль в конструкції. Її параметри є вихідними даними для оптимізації розподілу матеріалу в елементах конструкції, розміри поперечних перерізів яких відносно слабо впливають на загальну картину розподілу зусиль. Від параметрів силової схеми в значній мірі залежить загальна маса конструкції. Якщо силова схема обрана невдало, то навіть ретельне подальше проектування (в тому числі і параметричну оптимізація) не виправить її вад.

Між прагненням до підвищення міцності, твердості і стійкості конструкцій і вимог до зниження їх матеріаломісткості (маси або ваги) існує очевидна суперечність. Мистецтво проектування, що полягає в умінні створювати конструкції, які відповідають більшості з цих вимог, є важливим елементом інженерного мислення і в значній мірі визначає кваліфікацію проєктувальника.

Важливо звернути увагу на особливості цієї схеми, що складаються в тому, що процес проектування силових конструкцій є, як, втім, і сам процес проектування, нескінченним процесом. Це відбувається через те, що, найчастіше, оцінюючи варіанти силових схем, проектувальник порівнює їх тільки між собою, не знаючи меж можливого вдосконалення конструкції. Тому він ніколи не може бути впевнений в тому, що остаточно знайдене рішення є найкращим з можливих. Звичайно, він може керуватися досвідом попередніх розробок, але і до попередніх розробок в тій же мірі можуть бути віднесені такі ж міркування. Дуже рідко проектувальник має точні цифри можливих меж вдосконалення конструкції і тому йому доводиться в кінцевому підсумку зупинитися на якомусь своєму найкращому досягнутому результаті.

2.2. Постановка задачі

Постановка типового проектно-конструкторського завдання може виглядати так [7].

Задається розташування навантажень і закріплень, а також габарити плоскої проектної області, в яку повинна бути вписана конструкція. Потрібно знайти раціональну за критерієм маси (силової ваги) схему плоскої конструкції.

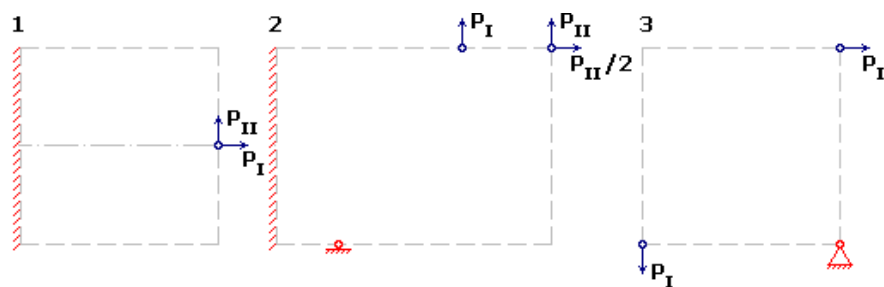


Рис. 2.2. Типові завдання для проектування силових схем конструкцій

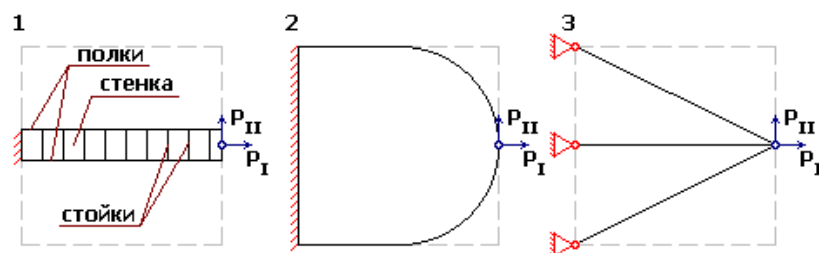


Рис. 2.3. Плоскі силові конструкції: 1. Балка. 2. Рама або пластина. 3. Ферма.

Деякі типові завдання на проєктування силових схем показані на рис. 2.2., Де P_I і P_{II} - дві різночасно діючі сили (тобто два випадки навантаження, два розрахункового випадку).

Проектне завдання в такій постановці розділяється на два етапи:

1-ий етап - відшукування раціональної структури конструкції (структурна оптимізація);

2-ий етап - відшукування в рамках прийнятої структури раціональних параметрів елементів конструкції (параметрична оптимізація).

Для таких проєктно-конструкторських завдань можна запропонувати велику кількість плоских силових конструкцій різного типу: балкових, рамних, ферменних, у вигляді пластин, або конструкцій змішаного типу. На рис. 2.3, наприклад, для першого (рис.2.2) завдання, наведені можливі варіанти виконання у вигляді балки, пластини і ферми.

Найпростішою з усіх типів конструкцій є ферма, яка за своєю побудовою і зображенню найбільш близька до поняття силової структури.

Як уже зазначалося, ферма була одним з перших об'єктів застосування оптимізаційних методів. Однак, незважаючи на простоту постановки та подання, завдання проєктування раціональних ферменних конструкцій, особливо конструкцій, що працюють на декілька розрахункових випадків, є вельми нетривіальними. Навіть для варіантів з малим числом закріплень і навантажень можна запропонувати досить велику кількість різноманітних структур різної складності, що представляє широке поле діяльності з генерування евристичних конструкцій.

Як проєктні змінні будемо використовувати площі поперечних перерізів стрижнів. Наведені вище міркування дозволяють вибрати для дослідження на початковій стадії підготовки до силового проєктування прості ферменні конструкції (плоский випадок).

Фермою називається конструкція (геометрично незмінна система), що складається з прямолінійних стрижнів, з'єднаних шарнірами на кінцях.

Під геометричною незмінюваністю розуміють здатність системи не допускати відносного переміщення своїх частин без їх деформації. Стрижні ферми працюють на розтягування і стиснення.

Основне призначення ферми складається в "передачі" впливів, яких докладають в деяких заданих точках, на опорні вузли.

Розглянуті зазвичай в теорії оптимального проектування завдання полягають в мінімізації кількості матеріалу (ваги ферми) за рахунок відповідного вибору положення вузлів ферми і площ поперечних перерізів її елементів. Реальна ферма має жорсткі вузли кріплення, тому вона є багаторазово статично невизначеної системою. У розрахунках зазвичай фігурує не реальна конструкція, а розрахункова схема. На рис. 2.4 представлені фермова конструкція (зліва) і розрахункова схема ферми (праворуч).

Розрахунковою схемою конструкції (в тому числі і ферменної) називається спрощене її уявлення (модель), яке фігурує в процесі розрахунку замість дійсної конструкції. Зокрема, дослідження різних ферменних конструкцій показали, що якщо жорсткі вузли ферми замінити ідеальними шарнірами, то це припущення незначно змінить значення зусиль в стержнях ферми.

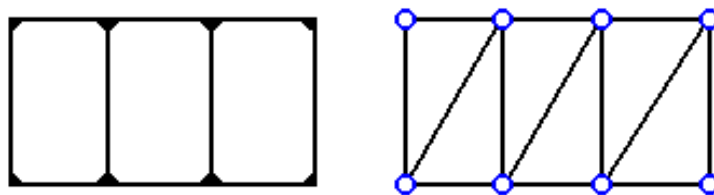


Рис. 2.4. Фермова конструкція (зліва) і розрахункова схема ферми (праворуч).

Тому за розрахункову схему ферменної конструкції приймають систему, у якій стрижні по кінцях з'єднані ідеальними шарнірами.

З огляду на те, що розрахункова схема визначає шляхи передачі сил від одного стержня до іншого, від однієї точки простору до іншої, розрахункову схему називають також і силовою.

Проектувальник, який розробляє подібну силову конструкцію, зазвичай обмежений розмірами простору, в якому він може розмістити ферму, обмежений він також і розташуванням можливих точок закріплення ферми і точок прикладання сил, можливі також інші обмеження.

Тому звичайне завдання проектування плоскої ферми може бути сформульоване таким чином:

Вважаються заданими:

- габарити проектної області (в нашому випадку прямокутної), в якій необхідно розмістити конструкцію;
- величини навантажень, їх напрямки, координати точок докладання зусиль;
- умови закріплення ферми: координати і типи опор, можливі місця розташування опор.

Потрібно в заданих габаритах проектної області знайти (спроектувати) раціональну схему плоскої ферменної конструкції і оптимальний розподіл матеріалу в ній.

Як вже говорилося, в процесі проектування силових конструкцій можна виділити два основних етапи:

- проектування структури, або структурна оптимізація. На цьому етапі відшукується силова схема або несуча структура конструкції, яка представляє собою основні шляхи передачі зусиль. При проектуванні силових схем на основі характеру навантаження і закріплення визначаються розташування і взаємозв'язок основних конструктивних елементів.

- параметрична оптимізація, в процесі якої проводиться відшукування найкращих параметрів елементів конструкції: призначення товщини елементів, підбір поперечних перерізів і т. п.

Задачі параметричної оптимізації в достатній мірі формалізовані і успішно вирішуються з використанням різних оптимізаційних методів. Значно менш формалізованими є завдання структурної оптимізації.

При евристичному вирішенні поставленого завдання силова схема конструкції вибирається проєктувальником на основі раніше набутих знань, накопиченого досвіду та інтуїції, що в принципі є нетривіальне завдання, особливо якщо для вихідної області поставлено кілька розрахункових випадків.

Саме тому на першому етапі проєктування силових конструкцій (проєктування структури) доцільно використовувати спеціальні критерії та спеціальні прийоми такі, як, наприклад, силова вага теоретично оптимальної конструкції (ТОК) і дослідження напружено-деформованого стану континуальної моделі, вписаної в задану проєктну область і відповідно навантажену.

Для отримання силової ваги ТОК в вихідну проєктну область вписується, а потім розраховується континуальна модель, що представляє собою ізотропну пластину постійної товщини при заданих умовах навантаження і закріплення. Істотна особливість такої моделі полягає в тому, що вона, в силу ізотропії властивостей, не визначає, як ферма, шляхи передачі зусиль в проєктній області.

Отже, для отримання силової структури конструкції в область, обмеженою зовнішніми розмірами проєктованої конструкції, вписується деяка континуальна модель, що включає в себе потенційно найбільше число можливих схем, після чого шукається оптимальний розподіл матеріалу в обраній моделі і знаходиться, таким чином, теоретично оптимальна конструкція.

Геометрична незмінність ферм

Числом ступенів свободи механічної системи називається кількість геометричних параметрів, які можуть змінюватися під час руху системи. Іншими словами, числом ступенів свободи називається кількість незалежних координат, які визначають положення системи при її переміщеннях.

Пристрої, що позбавляють систему однієї або декількох ступенів свободи, називаються зв'язком або зв'язками. Ферму можна розглядати як

механічну систему точок (вузли ферми), на яку накладено зв'язку (стрижні). Стрижень в плоских і просторових системах накладає один зв'язок, тобто позбавляє систему одного ступеня свободи. Число ступенів свободи для плоских ферм визначається формулою:

$$W = 2Y - C - K ,$$

де Y - число вузлів ферми (включаючи опори ферми), C - число стержнів ферми, K - число зв'язків, що накладаються опорами.

Якщо $W \leq 0$, то ферма геометрично незмінна і переміщення її частин не можуть відбуватися без деформації складових її елементів. В іншому випадку ферма - механізм і не може працювати як силова конструкція.

2.3. Структурна оптимізація. Вибір силової схеми ферменної конструкції

Під силовою схемою конструкції прийнято розуміти схему розташування основних силових елементів. При проектуванні силової схеми вважаються заданими обмеження на зовнішні розміри конструкції, навантаження, умови закріплення.

Евристичний пошук варіанту силової схеми ферменної конструкції.

При евристичному пошуку варіанту ферменної конструкції проектувальник спирається тільки на раніше набуті знання, накопичений досвід і інтуїцію, створюючи схеми конструкцій, в рамках обмежень поставленого завдання.

Аналітичне рішення задачі.

В основі аналітичного рішення задачі проектування конструкцій лежить метод силового аналізу. На його базі реалізується спільний підхід до проектування силових схем довільних конструкцій за таким планом:

- в область, обмежену зовнішніми розмірами проектованої конструкції, вписується деяка континуальна (суцільна) модель, що включає в себе потенційно найбільше число можливих схем.

- шукається оптимальний розподіл матеріалу в обраній моделі і знаходиться, таким чином, теоретично оптимальна конструкція (ТОК).

Тут використовується інтерактивний алгоритм оптимізації обрисів плоских силових конструкцій. Метою проектування є визначення таких обрисів конструкції, при яких вона при заданому матеріалі має мінімальну масу і задовольняє обмеженням по міцності. Суть алгоритму полягає в тому, що в проєктну область вписується однорідна ізотропна пластина постійної товщини з деякими певними механічними властивостями. Ця пластина розбивається мережею кінцевих елементів. Потім в алгоритмі використовується така властивість раціональних конструкцій, як рівноміцність або рівнонапруженість. Вирівнювання напруг проводиться за рахунок зміни товщини елементів пластини в ході ітераційного процесу, що складається з серії розрахунків напруженого стану КЕМ. Причому, всередині проєктної області можуть утворюватися порожнини, отримані після виродження елементів з нульовою товщиною, що мають напруги також близькі до нульових значень. Таким чином, в ході ітераційного процесу оптимізації виявляються обриси рівнонапруженої конструкції.

2.3.1. Метод силового аналізу

Формулювання і загальний план розв'язання задачі оптимізації обрисів плоских силових конструкцій може бути представлений наступним чином:

Вважається заданою деяка проєктна область S_0 , обмежена лінією ℓ_0 (див. рис. 2.5).

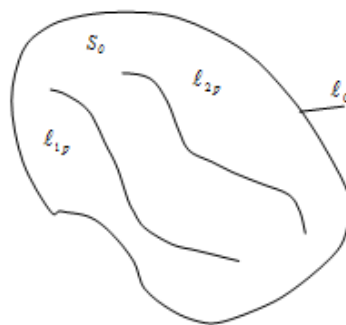


Рис. 2.5. Проєктна область

В середині області і на її межі можуть розташовуватися два сімейства ліній ℓ_{1p} і ℓ_{2q} , де $p = 1, 2, \dots, r, q = 1, 2, \dots, s$.

На ℓ_{1p} задані статичні умови (навантаження), на ℓ_{2q} - число випадків навантаження і кількість кінематичних умов відповідно.

Метою проектування є визначення таких обрисів конструкції, при яких вона при заданому матеріалі має мінімальну масу і задовольняє обмеженням по міцності.

Впишемо в проектну область S_0 однорідну ізотропну пластину постійної товщини δ_0 з деякими певними механічними властивостями. Розіб'ємо цю пластину мережею кінцевих елементів так, щоб внутрішні елементи не перетинали ліній ℓ_{1p} і ℓ_{2q} , а стикувалися між собою на цих лініях.

Тоді розглянуте вище завдання можна сформулювати в термінах математичного програмування, прийнявши в якості проектних змінних \bar{X} сукупність координат вузлів кінцево-елементної моделі за винятком вузлів, розташованих на зазначених лініях. Цільова функція залишається колишньою - це маса або об'єм матеріалу конструкції. Обмеження на величини проектних змінних визначаються умовами:

$$\bar{X} \in S_0, \bar{X} \notin l_{1p}, \bar{X} \in l_{2q},$$

а обмеження по міцності формуються у вигляді обмежень на верхній рівень напружень в КЕМ: $\sigma_{\max} \leq [\sigma]$,

де σ_{\max} - максимальне еквівалентне напруження, вибране з усіх елементів моделі; $[\sigma]$ - допустиме напруження матеріалу конструкції.

Таке формулювання завдань оптимізації дозволяє шукати її рішення за допомогою методів нелінійного математичного програмування. Однак велика кількість проектних змінних і значний обсяг обчислювальних витрат для розрахунку напружень при перевірці обмежень надзвичайно ускладнюють використання цих універсальних алгоритмів оптимізації. Більш прийнятні по обчислювальним витратам алгоритми оптимізації можна побудувати на основі таких відомих властивостей раціональних конструкцій, як рівномірність або

рівнонапруженість. Беручи ці властивості в якості критерію оптимальності, необхідно відзначити наступне: відомо, що рівномічні статично визначні конструкції при одному випадку навантаження є конструкціями мінімального обсягу. При декількох випадках навантаження рівномічні конструкції можуть і не бути конструкціями мінімального обсягу. Однак в більшості практичних задач вирівнювання напружень призводить або до конструкцій мінімального обсягу, або до конструкцій дуже близьким до них.

Вирівнювання напружень можна вести в ході ітераційного процесу, з визначенням напружено-деформованого стану КЕМ. Після кожного розрахунку проводиться зсув вузлів КЕМ в зону проходження основних силових потоків, тобто в сторону більш напружених ділянок. Причому, всередині проєктної області можуть утворюватися порожнини. Отримані після розриву вузлів, що мають напруги, близькі до нульових значень. Таким чином, в ході ітераційного процесу оптимізації виявляються обриси рівнонапруженість конструкції.

На базі методу силового аналізу реалізується наступний підхід до вибору силової схеми ферменної конструкції:

- в область, обмежену зовнішніми розмірами проєктованої ферменної конструкції, вписується деяка континуальна (суцільна) модель, що включає в себе потенційно найбільше число можливих силових схем;
- шукається оптимальний розподіл матеріалу в обраній моделі і знаходиться, таким чином, теоретично оптимальна конструкція (ТОК);
- аналізуються розподіл товщини, а також генеральні шляхи передачі зусиль в ТОК і з урахуванням конструктивних і технологічних вимог розробляються варіанти силових схем.

2.3.2. Оцінка ефективності силових схем

Природний критерій - вага конструкції, за яким традиційно оцінюється силова схема, має досить серйозні недоліки. Аби не заглиблюватися в їх перерахування, відзначимо основний з них, який отримує особливо важливе

значення для нашої задачі. Почнемо з того, що для розрахунку придуманої ферменної конструкції необхідно задати якісь значення площ поперечних перерізів стержнів ферми. Очевидно, що початкові значення площ F_i не будуть оптимальними і щоб остаточно судити про гідність запропонованої схеми необхідно проводити оптимізацію ферми, знаходячи найбільш раціональний розподіл матеріалу (значення F_i^{opt}). Таким чином, доводиться не тільки розробляти силову схему, але і знаходити найбільш оптимальний розподіл матеріалу в конструкції, тобто проводити параметричну оптимізацію, домагаючись мінімуму ваги конструкції при даній силовій схемі.

Але існують критерії, які дозволяють судити про гідність силової схеми після однократного розрахунку, не вдаючись до такого складного і трудомісткого процесу, як оптимізація. Як вже говорилося в попередньому розділі, одним з таких критеріїв є силова вага.

Силова вага - це кількісний показник, що характеризує величини і протяжність дій зусиль в конструкції. Для ферм силова вага: $S_G = \sum_{i=1}^n |N_i|_{\max} l_i$, де n - число стержнів ферми, l_i - довжина i -го стержня, $|N_i|_{\max}$ - максимальне зусилля в i -му стержні з усіх випадків навантаження.

Силова вага слабо залежить від розподілу матеріалу в конструкції. Її величина визначається, в основному, обраною силовою схемою. Ця властивість ваги дозволяє використовувати її в якості критерію ефективності при порівнянні різних силових схем - для кожного варіанту силової схеми досить задати будь-який довільний розподіл матеріалу (наприклад, прийняти площі поперечних перерізів стержнів ферми однаковими і рівними 1.0), виконати розрахунок напруженого стану конструкції і обчислити силову вагу. Менша величина цього критерію і визначить найбільш вигідний, з точки зору маси конструкції, варіант силової схеми.

Вельми важливо також не тільки оцінювати конкретні силові схеми ферменних конструкцій між собою, але і в порівнянні з оптимумом - нижньою межею величини силової ваги для заданих габаритів проектної області,

величин навантажень і розташування опор. Для цього порівнюється силова вага розробленої конструкції S_G і силова вага теоретично оптимальної конструкції $S_G^{ТОК}$. Очевидно, що завжди виконується умова $S_G^{ТОК} < S_G$.

Для отримання силової ваги ТОК в вихідну проєктну область вписується, а потім розраховується континуальна модель, що представляє собою ізотропну пластину постійної товщини при заданих умовах навантаження і закріплення. Істотна особливість такої моделі полягає в тому, що вона, в силу ізотропії властивостей, не визначає, як ферма, шляхи передачі зусиль в проєктній області.

Для отримання силової структури конструкції в область, обмежену зовнішніми розмірами проєктованої конструкції, також вписується деяка континуальна модель, що включає в себе потенційно найбільше число можливих схем, після чого шукається оптимальний розподіл матеріалу в обраній моделі і знаходиться таким чином теоретично оптимальна конструкція (ТОК).

2.3.3. Розрахунок силової ваги теоретично оптимальної конструкції

Силовa вага - це кількісний показник, що характеризує величини і протяжність дії зусиль в конструкції. При розрахунку силової ваги ТОК проводиться одноразовий розрахунок напружено-деформованого стану пластини, після якого підраховується силова вага:, де $S_G = \int_V \sigma_{\text{екв}}^{\text{max}} dV$, де

$\sigma_{\text{екв}}^{\text{max}}$ — максимальне з усіх випадків навантаження еквівалентне напруження, підраховане по четвертій теорії міцності (теорії енергії формозміни):

$$\sigma_{\text{екв}} = \sqrt{(\sigma_x^2 + \sigma_y^2 - \sigma_x \sigma_y + 3\tau_{xy}^2)}.$$

Обчислена таким чином силова вага пластини з достатньою точністю (до 5-7%) дає прогноз величини силової ваги ТОК і, отже, може служити нижньою межею, до якої необхідно прагнути при виборі структури ферменної конструкції.

2.4. Розрахунок напружено-деформованого стану

Розрахунок напружено-деформованого стану полягає у визначенні переміщень точок елементів конструкції і значень напруг, що виникають в елементах конструкції при навантаженні. Для вирішення цього завдання в розпорядженні проєктувальника є безліч методів, найпростіші з яких розглядаються в курсі «Опір матеріалів». Але, як вже говорилося, в даний час метод кінцевих елементів є найбільш поширеним, що пояснюється його перевагами обчислювального характеру. Тому використання його в практиці проєктування ферменних конструкцій є цілком природним, тим більше що цей же метод можна застосовувати і при розрахунку пластин для визначення напружено-деформованого стану та силової ваги ТОК. Покажемо найпростіші варіанти використання МСЕ для плоских ферм.

2.4.1. Розрахунок напружено-деформованого стану ферменної конструкції

Розрахунок напружено-деформованого стану ферменної конструкції за методом скінчених елементів проводять наступним чином.

Для моделювання ферм використовують лінійний скінчений елемент - прямий стрижень з двома вузлами і постійним поперечним перерізом (рис. 2.6).

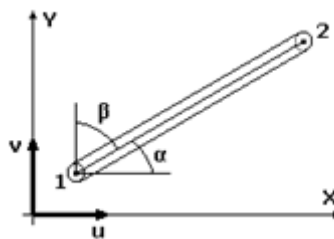


Рис. 2.6. Стрижневий скінчений елемент

Стрижень працює на розтягування і стиснення і має по два ступені свободи u і v в кожному вузлі. Під ступенем свободи вузла розуміється переміщення вузла щодо глобальної системи координат. Ступені свободи вузлів відповідають невідомим в матриці рівнянь.

Розмір матриці визначається кількістю задіяних елементами вузлів, помножених на кількість ступенів свободи вузлів; при цьому з кількості ступенів свободи вузлів віднімається число граничних умов.

Для ферменної конструкції вузли - це шарніри, що з'єднують стрижні конструкції, а самі стрижні і є кінцеві елементи.

Якщо подумки виділити кінцевий елемент і спробувати переміщати його вузли відносно один одного, то, очевидно, кінцевий елемент буде деформуватися і в ньому виникнуть напруги. Так як деформація одного елемента не може не супроводжуватися деформацією сусідніх елементів (переміщуваний вузол належить і сусіднім елементам), то при навантаженні конструкції переміщення її точок (вузлів) викликають зміни напруженого стану у всіх кінцевих елементах. Навантажена конструкція, деформуючись, займе якийсь стан. Це положення відповідає умові (а, отже, рівняння) рівноваги. Рішення рівняння рівноваги дає нам значення зсувів (переміщень) вузлів, а через них - напружень в елементах.

Матриця жорсткості стрижневого СЕ:

$$K = \frac{EF}{l} \begin{bmatrix} d & -d \\ -d & d \end{bmatrix}, \text{ де } E - \text{модуль пружності матеріалу ферми;}$$

F - площа поперечного перерізу стержня;

l - довжина стрижня;

$$d = \begin{bmatrix} \cos^2 \alpha & \cos \alpha \cos \beta \\ \cos \alpha \cos \beta & \cos^2 \beta \end{bmatrix} - \text{матриця напрямних косинусів.}$$

Для вирішення підсумкової системи лінійних алгебраїчних рівнянь:

$KU = P$, (де K - матриця жорсткості; U - вектор вузлових переміщень; P - вектор вузлових навантажень;) як правило, застосовується метод Холецкого, що складається з трьох етапів:

На першому етапі з рівності $G^T G = K$ визначається трикутна матриця G .

На другому - вирішується перша допоміжна система рівнянь (прямий хід) $G^T X = P$..

На третьому етапі вирішується друга допоміжна система (зворотний хід) $GU = X$, з якої визначається шуканий вектор U .

Розрахунок може враховувати, що частина стрижнів виявляється стислими, а може проводитися і без урахування стиснення. Очевидно, що стиснений стержень може втратити стійкість задовго до того, як буде досягнуто граничне значення напруг $[\sigma]$, яке може витримувати матеріал конструкції, так зване допустиме напруження. Тому стислі стрижні доводиться робити з більшою площею поперечного перерізу. При цьому ще величезне значення має форма поперечного перерізу, так як від форми залежить момент інерції перерізу, який входить ваговою складовою в формулу визначення критичної сили втрати стійкості для стисненого стержня:

$$P_{кр} = \frac{\pi^2 EJ}{\ell^2}, \text{ де } J - \text{мінімальний момент інерції поперечного перерізу стержня,}$$

E - модуль пружності матеріалу, ℓ - довжина стрижня. Це - так звана формула Ейлера для стисненого стержня з шарнірно-опертими кінцями, вперше отримана академіком Петербурзької академії наук Л. Ейлером в 1744р. Облік явища втрати стійкості необхідна складова розрахунку ферм.

2.5. Висновки до розділу

В якості основного об'єкта розробки виступає фермова конструкція. В даному розділі розглянуті евристичний підхід і підхід на основі методу силового аналізу. У розділі розглянуто методику проектування силових схем ферменних конструкцій методом силового аналізу і розрахунок напружено-деформованого стану ферменної конструкції.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМИ ДЛЯ ПРОЄКТУВАННЯ І РОЗРАХУНКУ БУДІВЕЛЬНИХ КОНСТРУКЦІЙ

3.1. Обґрунтування вибору середовища розробки системи проєктування і розрахунку будівельних конструкцій

Delphi - останнє досягнення на ниві візуального програмування. Цілою низкою дуже серйозних видань вона визнана продуктом вищої якості, неодноразово нагороджена різноманітними нагородами, сотні тисяч розробників і звичайних користувачів одноголосно вибирають цю систему програмування для створення власних додатків.

Призначення інтегрованого середовища Delphi:

- розробка багатовіконних призначених для користувача додатків;
- створення багатофункціональних систем загального призначення;
- проєктування баз даних будь-якої складності і засобів управління БД;
- розробка систем обробки текстової, графічної, відеоінформації та звуку;
- створення графічної операційної оболонки;
- написання прикладних програм і бібліотек динамічного компонування;
- створення одно- і багатокористувацьких інтерфейсів;
- розробка мережових додатків;
- написання програм з використанням засобів Internet;
- і багато чого іншого.

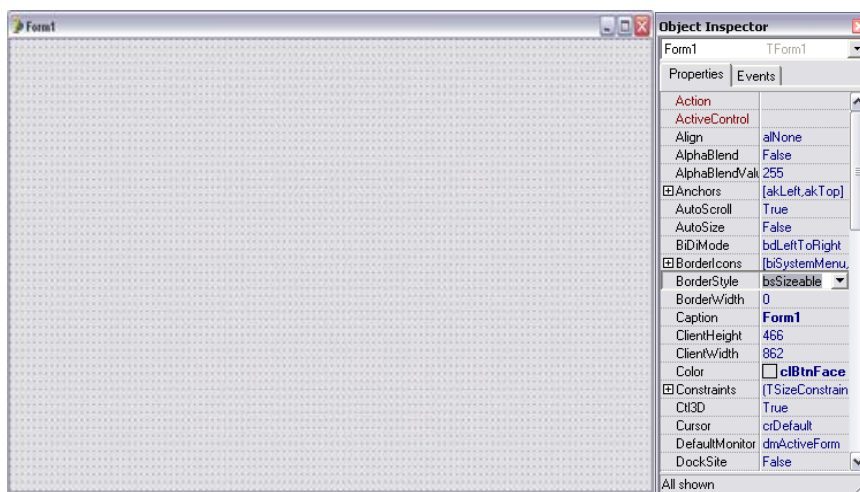


Рис. 3.1. Головна форма і інспектор об'єктів

Delphi - це синтез декількох ключових технологій:

- Ефективний компілятор в машинний код.
- Об'єктно-орієнтована модель компонент.
- Візуальне конструювання додатків з програмних прототипів.
- Розширені засоби для створення та управління базами даних.

Delphi - це система програмування, яка базується на мові програмування (Object Pascal), має свій редактор, компілятор і відладчик. Написання програми на Delphi зводиться до компонуванні на екрані об'єктів, які мають певну графічну інтерпретацію, і підключенню рядків коду, як і в програмі на будь-якій іншій мові. Інакше кажучи, Delphi просто реалізує візуальну концепцію програмування.

Отже, Delphi - це не лише текстовий редактор та компілятор. Це потужне середовище розробки, яке значно спрощує процес створення програмних додатків. Засоби інтерфейсу програмування застосунків (API) дозволяють створювати утиліти, які легко інтегруються у середовище розробки Delphi.

Середовище Delphi включає повний комплект візуальних інструментів для швидкої розробки додатків (RAD - rapid application development), підтримує створення користувацького інтерфейсу та забезпечує можливість підключення до корпоративних баз даних. VCL - це бібліотека візуальних компонент, яка включає стандартні об'єкти для створення користувацького інтерфейсу, об'єкти управління даними, графічні та мультимедійні об'єкти, діалогові вікна та об'єкти для управління файлами, а також функції для управління DDE і OLE.

Легко помітити, що елементи екрана, які представляють додатки Windows, досить прості. Компоненти зберігаються в бібліотеці компонентів, яка містить всі об'єкти, необхідні для створення повноцінних програм, які використовують інтерфейс Windows.

Об'єктно-орієнтована природа Delphi робить бібліотеку компонентів надзвичайно гнучкою. Якщо об'єкту потрібна додаткова функціональність або потрібно модифікувати поведінку компонента, можна успадковувати новий

компонент з того, що вже зберігалось в бібліотеці, і додати йому нових властивостей.

Тепер, коли програмування стало полягати в простому маніпулюванні компонентами і об'єктами, з'являються шаблони, які навіть це завдання роблять тривіальним. Delphi оперує чотирма типами шаблонів: формами, доповненнями, компонентами і кодами. Шаблони форми, доповнення та компонент дають можливість повторно використовувати створені раніше колекції об'єктів або в окремих програмах, або як основу для нової програми. Шаблон коду - це новий засіб, який значно зменшує потреби у введенні повторюваних фрагментів коду.

Використання стовідсоткової компіляції дає ще одну перевагу, яка полягає в створенні бібліотек динамічного компонування (DDL), які можуть містити будь-які компоненти з бібліотеки компонентів. Потім ці бібліотеки можна використовувати у власних додатках Delphi чи поширювати як незалежні компоненти для інших програм.

У Delphi зручно реалізувати красивий інтерфейс програми додатку завдяки різноманітності стилів реалізації (колірна палітра, інтерфейс, структура ЕП, спосіб подачі матеріалу).

Головною відмінною рисою і перевагою даного виду програмування є простота використання. Користувач вибирає необхідні компоненти з представленої палітри заготовок, що значно скорочує час роботи і рятує від виснажливого програмування. Delphi подбала і про шаблони коду, щоб уникнути множинних помилок. Таким чином, користувач вибирає необхідні йому компоненти, заносить їх в свою форму, визначає їх властивості за допомогою інспектора об'єктів і заповнює шаблони коду необхідними командами в оброблювачі подій.

Справжня модифікація системи реалізована на РС для роботи під управлінням ОС Microsoft® Windows 10 в системі програмування для Windows - Borland Delphi 7 і Visual Fortran.

Мова Visual Fortran була розроблена як розширення Fortran 90, адаптоване до сімейства ОС Windows. Вона дозволила компілювати DLL-бібліотеки для полегшення використання і підключення старих напрацювань в області машинних обчислень. Мова Fortran має довгу історію, і зарекомендувала себе як одна з кращих для проведення математичних розрахунків в різних областях науки і техніки. За довгі роки було розроблено, налагоджено і перевірено безліч обчислювальних бібліотек і пакетів, придатних для найрізноманітніших завдань, які доцільно використовувати і продовжувати їх розвиток, використовуючи накопичений досвід попередніх поколінь програмістів. Використовуючи сучасні засоби розробки ПЗ, що надаються користувачам, можна домогтися гарних показників продуктивності і працездатності систем, застосовуючи накопичений за роки багаж напрацювань.

Виходячи з принципів використання САПР "Проектування і розрахунок будівельних конструкцій", які передбачають можливість модифікації системи, система реалізована як MDI додаток.

Термін MDI (Multiple Document Interface) дослівно означає багато документальний інтерфейс і описує додатки, здатні завантажити і використовувати одночасно кілька документів і об'єктів. Зазвичай MDI-додатки складаються мінімум з двох форм - батьківської і дочірньої. Батьківська форма служить контейнером, що містить дочірні форми, які укладені в клієнтську область і можуть переміщатися, змінювати розміри, мінімізуватися або максимізуватися. У додатку до САПР "Проектування і розрахунок будівельних конструкцій" використовується дочірня форма: проєкт "ферменна конструкція".

Деякі форми (наприклад, результати розрахунків) реалізовані як Modal Form, тобто модальні форми, при появі яких забороняються всі дії поза цією формою всередині програми, що дуже зручно для уникнення можливих помилок і неправильної інтерпретації деяких дій і даних.

Обмін інформацією в системі відбувається через звичайні текстові файли (ASCII-файли). При роботі з системою, для проєкту фермена конструкція - при збереженні на диск створюється файл з відповідним форматом. Результати різних розрахунків також записуються в відповідні файли.

При реалізації системи була застосована також модель програмування на основі DLL, яка дозволяє добитися модульності деяких елементів системи. Через DLL реалізовані розрахункові та оптимізаційні методи. DLL (Dynamic Link Library - динамічна компонована бібліотека) - це окремо компільовані, що виконуються бібліотеки, що підключаються до програми або системи під час виконання. Зручність їх застосування полягає в можливості зосередити в одному модулі фрагменти коду, даних і ресурсів, які потім можуть використовуватися відразу декількома програмами. Використання DLL зменшує обсяг EXE-файлу за рахунок розміщення частини коду в бібліотеках. Модифікація DLL (наприклад, налагодження) призводить до зміни файлів або програм, в яких вона використовується. DLL можна при необхідності завантажувати і потім вивантажувати з пам'яті. Це дає програмі змогу в будь-який момент регулювати наявність пам'яті і інших ресурсів.

3.2. Загальна структура і функціональна схема системи проєктування і розрахунок будівельних конструкцій

Система проєктування і розрахунок будівельних конструкцій призначена для розрахунку і оптимізації плоских ферменних конструкцій.

За допомогою системи можна вирішувати такі завдання:

- Проєктування силової схеми.

Заданими вважаються навантаження, умови закріплення та межі області прямокутної форми, в яку повинна бути вписана плоска ферма. Потрібно знайти раціональну за критерієм маси силову схему (структуру) ферменної конструкції.

- Розрахунок ферми.

Заданими вважаються силова схема і розподіл матеріалу в ферменній конструкції, навантаження, умови закріплення. Потрібно визначити переміщення вузлів, напруження і зусилля в стержнях ферми, а також необхідні з умов міцності і стійкості площі поперечного перерізу стержнів і мінімальні моменти інерції.

Загальна послідовність проєктних робіт представлена на рис. 3.2.

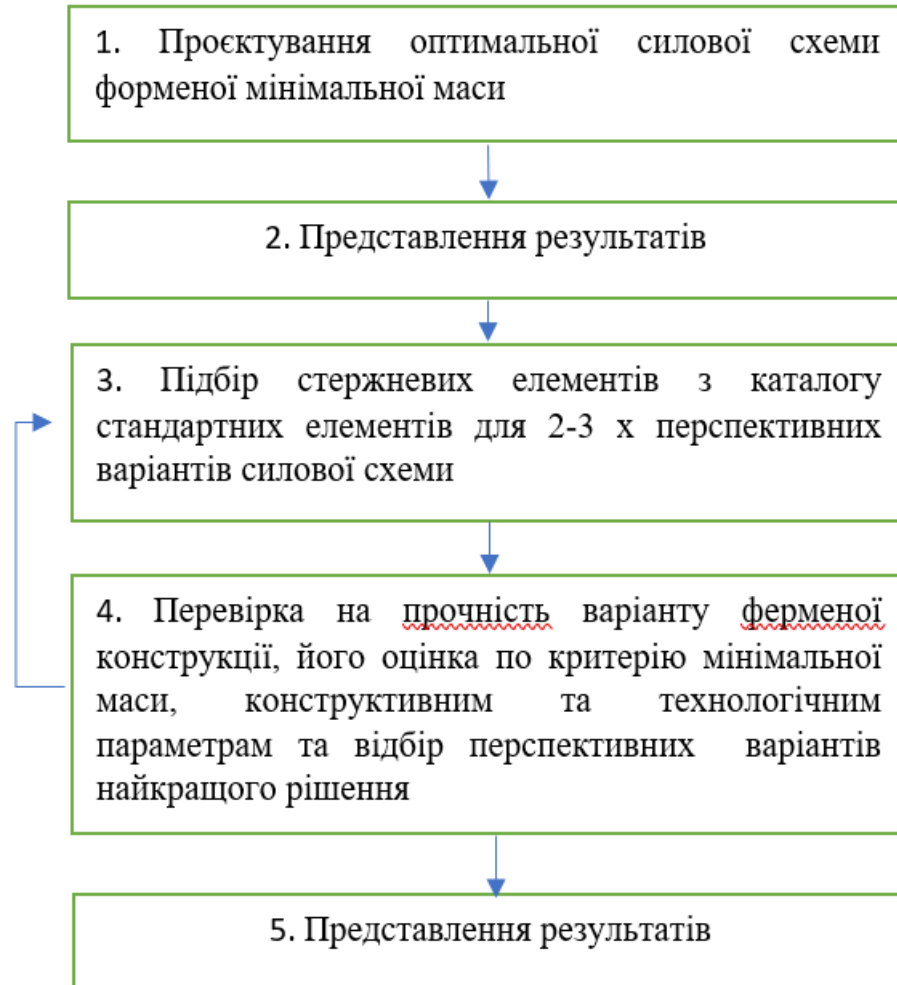


Рис. 3.2. Загальна послідовність проєктних робіт

- Формулювання проєктно-конструкторського завдання по розробці ферменної конструкції в системі проєктування плоских конструкцій виглядає наступним чином:

Вважаються заданими навантаження, умови закріплення, а також габарити проєктної області, в якій необхідно розмістити і спроектувати плоску ферменну конструкцію наприклад рис. 3.3.

Тут задані габарити прямокутної області (a і b), координати і вид можливих опор майбутньої ферми, а також навантаження на ферму у вигляді зосереджених сил P_I і P_{II} , що діють на конструкцію.

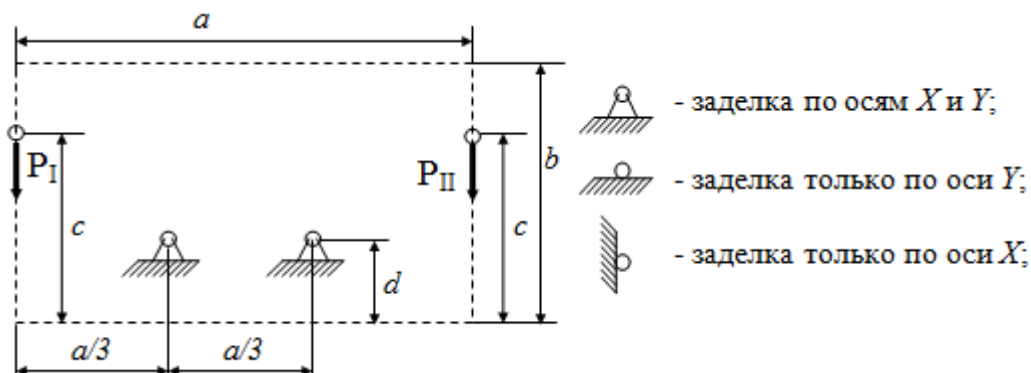


Рис. 3.3. Приклад умови на розрахунок

Потрібно в заданих габаритах знайти раціональну силову схему плоскої ферменої конструкції.

Зрозуміло, що для даного прикладу можна запропонувати безліч плоских ферменних конструкцій, дві з яких наведені на рис. 3.4.

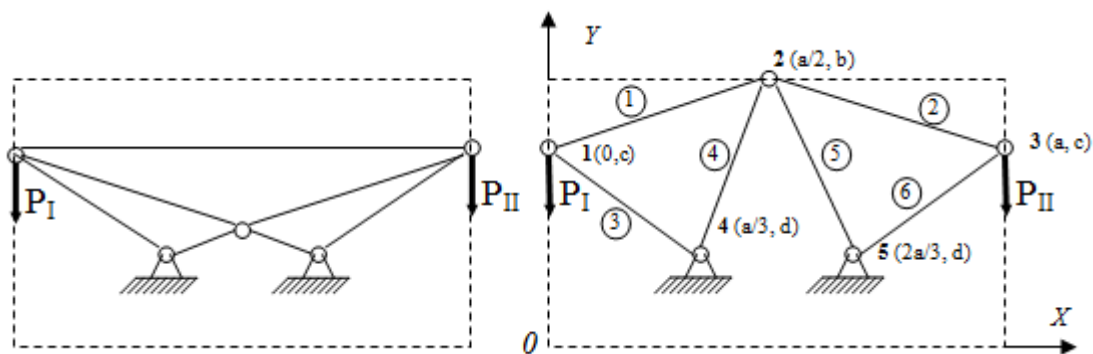


Рис. 3.4. Дві плоскі ферменні конструкції

Необхідно запропонувати таку конструкцію, яка б виділялася з усіх можливих за будь-яким критерієм. Традиційним критерієм в авіаційно-ракетній техніці є вага. Саме за цим критерієм в системі проєктування плоских конструкцій в кінцевому підсумку оцінюються всі схеми плоских ферменних конструкцій.

Стрижнева система називається шарнірної фермою, якщо її елементами є прямі стрижні, що працюють тільки на розтяг-стиск; її вузли і опори - шарнірні; вона - геометрично незмінна. При цьому навантаження на ферму

прикладається тільки в вузлах у вигляді зосереджених сил. Ферма називається плоскою, якщо осі всіх стрижнів і зовнішні сили лежать в одній площині, яка одночасно є площиною геометричній і масової симетрії абсолютно твердих тіл.

Так як по стрижнях ферми передаються зусилля, то схеми ферменних конструкцій (та й не тільки ферменних) прийнято називати силовими схемами. Таким чином, силова схема конструкції - це схема розташування основних силових елементів, які зумовлюють генеральні шляхи передачі зусиль, а головним завданням проєктанта є розробити (придумати, спроектувати) саму раціональну силову схему конструкції і дати свої рекомендації по найбільш оптимальному розподілу матеріалу в ній.

Вага або об'єм матеріалу конструкції залежить від багатьох параметрів: виду і інтенсивності діючого навантаження, виду та кількості використовуваних опор, кількості і довжини стрижнів, площі поперечних перерізів стрижнів, характеристик матеріалу конструкції.

Обмежившись, наприклад, рамками завдання оптимізації розподілу матеріалу в конструкції при заданій силовій схемі, коли відомі навантаження, матеріал, довжини і кількість стрижнів, які утворюють ферму, а також координати вузлів ферми, ми отримаємо задачу пошуку мінімуму функції $G(F_i)$ багатьох змінних з обмеженнями у вигляді нерівностей. В якості змінних тут виступають невідомі площі поперечних перерізів стрижнів F_i (де i - число стрижнів ферми), в якості нерівностей - обмеження по міцності і стійкості, а також габаритні і технологічні обмеження. Неприємною особливістю цієї функції (при цьому, не тільки для ферми) є її багатоекстремальність, тобто наявність великої кількості локальних мінімумів, що вельми ускладнює математичне рішення задачі.

Розгляд поставленого завдання проєктування плоских конструкцій проводиться з урахуванням того, що є підсистема "Проєктування і розрахунок будівельних конструкцій".

На рисунку 3.5. представлена схема системи проектування плоских конструкцій.

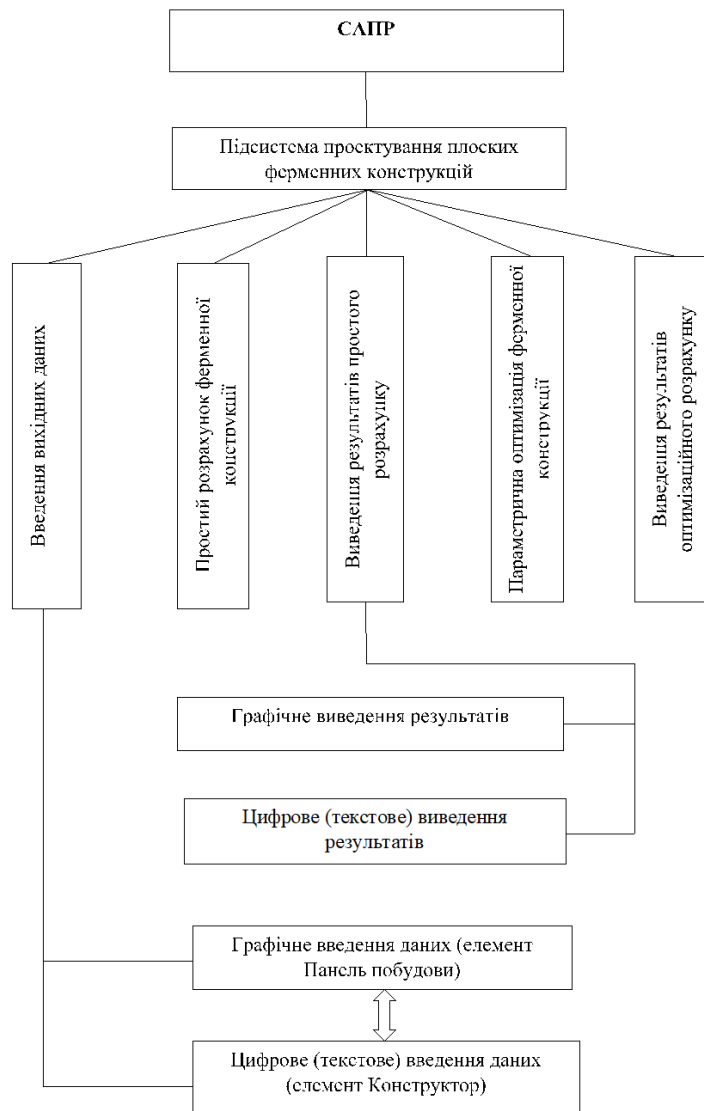


Рис. 3.5. Схема системи проектування плоских конструкцій


3.3. Опис використання підсистеми "ферменна конструкція" в системі проектування і розрахунку будівельних конструкцій

3.3.1. Загальні відомості

Створення нового проєкту

В меню Проєкт виберіть команду Створити, а потім тип створюваного проєкту: фермена конструкція.

Також, можна вибрати тип створюваного проєкту зі спливаючого меню, що з'являється при натисканні правої кнопки миші на головному вікні системи.


Щоб створити новий проєкт з використанням за замовчуванням типом, натисніть кнопку Створити .

Збереження проєкту

У системі передбачена можливість збереження активного проєкту, тобто проєкту, з яким зараз ведеться робота, незалежно від того, чи є він новоствореним чи ні.

Збереження нового проєкту

1. Натисніть кнопку Зберегти .

2. Щоб зберегти документ в іншій теці, виберіть потрібний диск зі списку Тека або клацніть двічі потрібну папку в списку папок. Щоб зберегти документ у новій папці, натисніть кнопку Створити теку .

3. Введіть ім'я проєкту в поле Ім'я файлу.

4. Натисніть кнопку Зберегти.

Збереження існуючого проєкту

Натисніть кнопку Зберегти .

Збереження копії проєкту


1. Відкрийте проєкт, для якого слід створити копію.

2. Виберіть команду Зберегти як в меню Проєкт.


3. У поле Ім'я файлу введіть нове ім'я файлу.

4. Натисніть кнопку Зберегти.

Закриття проєкту

Натисніть кнопку Закрити  в системному меню вікна поточного проєкту.

Вихід з системи

Натисніть кнопку Закрити  в системному меню головного вікна системи або виберіть команду Вихід в меню Проєкт, або натисніть клавіші [ALT] + [X].

Якщо Вами в процесі роботи в системі створювалися файли (зберігалися) поза робочим каталогом, то для їх видалення, необхідно задати пошук файлів з наступними розширеннями:

- * .frm - Файли з вихідними даними ферм;
 - * .vuv - Файли з результатом простого розрахунку ферм;
 - * .vuf - Файли з результатом параметричної оптимізації ферм;
- і видалити знайдені файли.

3.3.2. Початок роботи

Після створення нового проєкту "фермена конструкція" (або відкриття існуючого проєкту) ми маємо на екрані наступну панель:

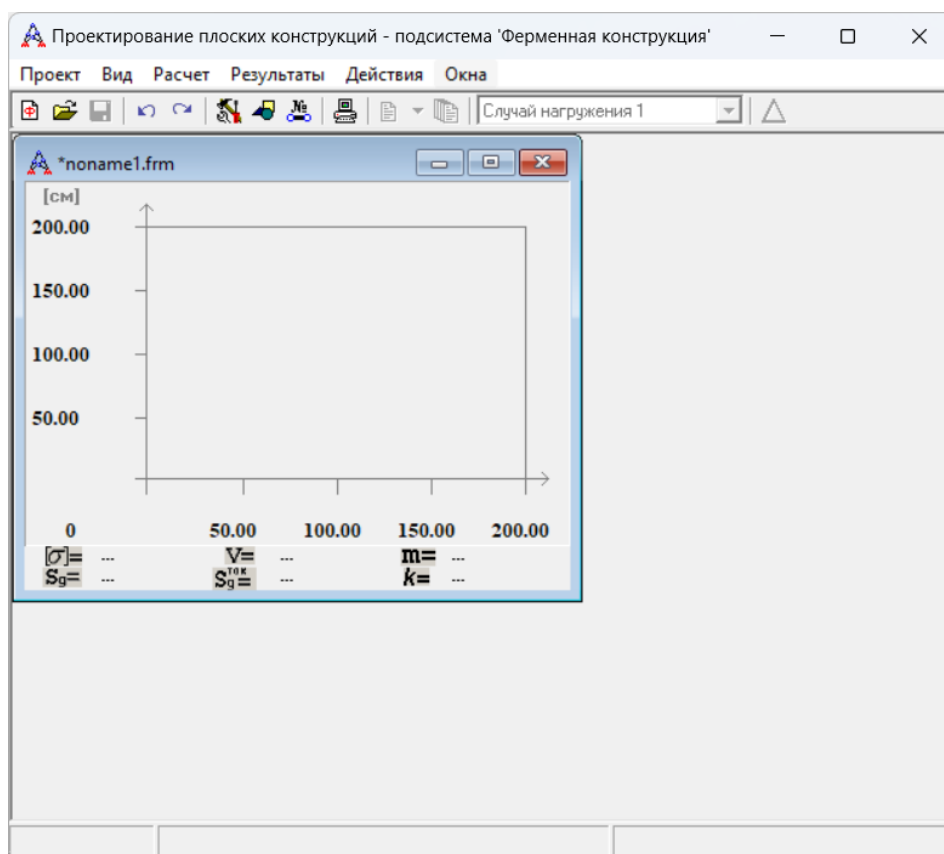


Рис. 3.6. Інтерфейс проєкту "ферменна конструкція"

тут:

- **вікно проєкту** (робоча область проєкту), де власне відображається проєктована ферменна конструкція; у вікні проєкту також показані габарити проєктної області (прямокутної), в межах якої працюємо (розміщуємо ферменну конструкцію) і одиниці лінійної розмірності;

- **панель інструментів**, за допомогою функцій якої відбувається робота над проєктом;

- **меню** - доповнюється пунктами, доступ до яких можливий при роботі з даним типом проєкту;

При роботі з проєктом передбачаються наступні режими роботи:

- Побудова ферменної конструкції (введення, редагування, видалення будь-яких даних на будь-якому етапі для зручного, простого і швидкого створення необхідної конструкції)

- Розрахунок ферменної конструкції, що включає:

- а) простий розрахунок ферменної конструкції;

- б) параметрична оптимізація ферменної конструкції.

- Перегляд і аналіз результатів, отриманих після розрахунку ферменної конструкції (в текстовому або графічному поданні), що включають:

- а) результати простого розрахунку;

- б) результати оптимізаційного розрахунку;


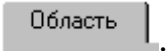
- в) графічне представлення результатів простого розрахунку.

3.3.3. Побудова ферменної конструкції

Розмірності, матеріали


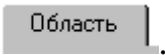
Побудова ферменної конструкції має на увазі завдання певного набору даних, достатнього для проведення простого або оптимізаційного розрахунку: розмір проєктної області, матеріал, розмірність нашого завдання, топологія конструкції і т. д.

Завдання розмірностей розв'язуваної задачі

Перед завданням основного масиву даних, необхідно вибрати одиниці виміру. Для вирішення нашої задачі задаються лінійна розмірність і розмірність сил. Для цього необхідно відобразити на екрані елемент системи "Конструктор", натиснувши кнопку Конструктор  і вибрати закладку Область .


У розділі Розмірність завдання виберіть зі списків Лінійна розмірність (мм - міліметри, см - сантиметри, М - метри) і Розмірність сил (Н - ньютони, кг - кілограм-сили) одиниці виміру.

Завдання матеріалу конструкції


Для завдання матеріалу необхідно відобразити на екран елемент системи "Конструктор", натиснувши кнопку Конструктор  і вибрати закладку Область .

У розділі Фізичні умови виберіть зі списку матеріал з необхідними Вам характеристиками: модулем пружності E і допущеною напругою $[\sigma]$, які наведені до обраних одиниць вимірювань.

Додавання матеріалу


Якщо відповідного матеріалу не знайдено (за замовчуванням після інсталяції системи для типу проєкту "ферменна конструкція" в списку матеріалів доступні Сталь і Алюміній (Алюмінієвий сплав)), то можливо створити новий матеріал для даного типу проєкту, натиснувши кнопку Додати .. У вікні необхідно задати ім'я матеріалу (не більше 20 символів), модуль пружності і напругу, що допускається, наведені до обраних одиниць вимірювань. Переходи між полями введення здійснюються натисканням клавіші <Tab> або за допомогою миші. Для створення матеріалу натисніть кнопку Додати або клавішу <Enter>.

Видалення матеріалу

Якщо матеріал був заданий невірно або потрібно видалити який-небудь матеріал, то вибравши його зі списку матеріалів, натисніть кнопку Видалити .

Вкладені в систему матеріали (за замовчуванням): Сталь, Алюміній (Алюмінієвий сплав) видаленню не підлягають (їх видалити неможливо).

Використання елемента системи "Панель побудови" для завдання ферменної конструкції

Щоб відобразити на екран елемент системи "Панель побудови" натисніть кнопку Панель побудови  на панелі інструментів проекту.

За допомогою функцій елемента системи "Панель побудови" можливо графічне створення ферми безпосередньо у вікні проекту.

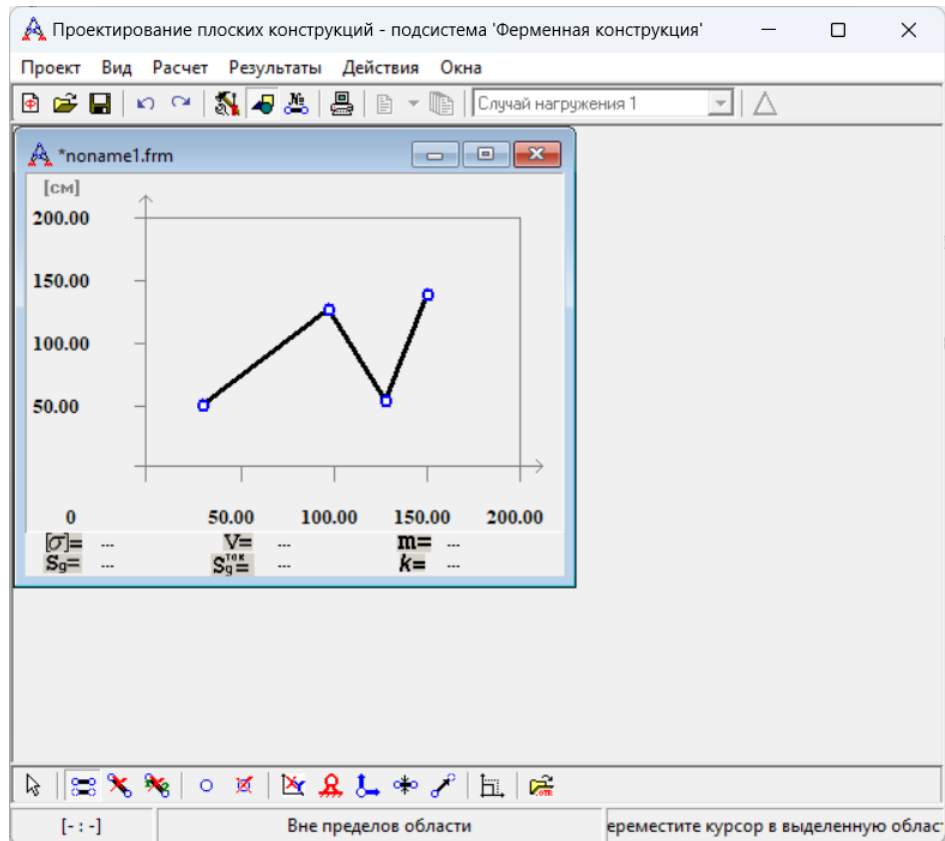



Рис. 3.7. Використання елемента системи "Панель побудови"

Функції панелі побудови

- Зміна габаритів проектної області;
- Малювання вузлів;
- Малювання стрижнів;
- Видалення вузлів;
- Видалення стрижнів;
- Завдання типу закріплення вузлів;
- Завдання виду навантаження вузлів;
- Коригування координат вузлів;
- Зміна площі перерізу стержнів.
- Відкриття фону силових ліній з проекту «Пластина»

При роботі над фермою через "Панель побудови", можна використовувати елемент системи "Конструктор", як паралельну гілку для перевірки, аналізу та коригування даних, що вводяться в цифровому (текстовому) вигляді.

Зміна габаритів проєктної області

Щоб змінити габарити проєктної області натисніть кнопку Розмір області  на панелі побудови.

У вікні задайте розміри області по X і Y, не порушуючи запропонованих в цьому вікні діапазонів зміни розмірів області. Ввівши необхідні розміри, натисніть кнопку ОК або клавішу <Enter>.

Доступні розміри габаритів проєктної області якщо:


- не заданий жоден елемент конструкції (новий проєкт)

Доступні розміри габаритів проєктної області в системі від 0 до 9999.99 (включно) по X і Y, обраних лінійних одиниць (лінійної розмірності), з урахуванням формату дрібних чисел x.XX (два десяткових знаки після коми).

- заданий будь-який елемент конструкції


Доступні розміри габаритів проєктної області в системі від максимальних координат X і Y одного з вузлів до 9999.99 (включно) по X і Y, обраних лінійних одиниць (лінійної розмірності), з урахуванням формату дрібних чисел x.XX (два десяткових знаки після коми).

Малювання вузлів

Щоб намалювати вузол (додати) з певними координатами натисніть кнопку Додати вузол  на панелі побудови. Ви увійшли в режим малювання вузлів.

- перемістіть курсор миші у виділену область (прямокутна проєктна область) у вікні проєкту;
- переміщайте курсор усередині неї поки не досягнете необхідних вам координат (поточні координати відображаються в поле координат рядка стану) і натисніть ліву або праву кнопку миші;
- у вікні проєкту з'явиться створений вами вузол.

Малювання стрижнів

Щоб намалювати стрижень (додати) натисніть кнопку **Додати стрижень**  на панелі побудови. Ви увійшли в режим малювання стрижнів.

Можливо кілька варіантів малювання стержня:

Створення стержня з двома новими вузлами

- перемістіть курсор миші у виділену область (прямокутна проєктна область) у вікні проєкту;
- переміщайте курсор усередині неї, поки не досягнете необхідних вам координат першого (початкового) вузла стержня (поточні координати відображаються в поле координат рядка стану) і натисніть ліву кнопку миші;
- не відпускаючи кнопку миші (метод "Гумові нитки"), перемістивши курсор всередині області поки не досягнете необхідних вам координат другого (кінцевого) вузла стержня (поточні координати відображаються в поле координат рядка стану) і відпустіть кнопку миші;
- у вікні проєкту з'явиться створений вами стрижень з двома новими вузлами.

Створення стрижня, використовуючи вже наявні вузли:

а) використання існуючого вузла в якості початкового

- перемістіть курсор миші у виділену область (прямокутна проєктна область) у вікні проєкту;
- перемістіть курсор на вузол, який ви хочете використовувати як перший (початковий) вузол стрижня (прив'язка (захоплений вузол) до вузла відображається в інформаційному полі рядка стану) і натисніть ліву кнопку миші;
- не відпускаючи клавішу миші (метод "Гумові нитки") перемістіть курсор усередині неї, поки не досягнете необхідних вам координат другого (кінцевого) вузла стержня (поточні координати відображаються в поле координат рядка стану) і відпустіть кнопку миші;
- у вікні проєкту з'явиться створений вами стрижень з новим вузлом.

б) використання існуючого вузла в якості кінцевого

- перемістіть курсор миші у виділену область (прямокутна проектна область) у вікні проекту;

- переміщайте курсор усередині неї, поки не досягнете необхідних вам координат першого (початкового) вузла стержня (поточні координати відображаються в поле координат рядка стану) і натисніть ліву кнопку миші;

- не відпускаючи клавішу миші (метод "Гумові нитки") наведіть курсор на вузол, який ви хочете використовувати як другий (кінцевий) вузол стрижня (прив'язка (захоплений вузол) до вузла відображається в інформаційному полі рядка стану) і відпустіть кнопку миші;

- у вікні проекту з'явиться створений вами стрижень з новим вузлом.

в) використання існуючих вузлів в якості початкового і кінцевого


- перемістіть курсор миші у виділену область (прямокутна проектна область) у вікні проекту;

- перемістіть курсор на вузол, який ви хочете використовувати як перший (початковий) вузол стрижня (прив'язка (захоплений вузол) до вузла відображається в інформаційному полі рядка стану) і натисніть ліву кнопку миші;

- не відпускаючи клавішу миші (метод "Гумові нитки") наведіть курсор на вузол, який ви хочете використовувати як другий (кінцевий) вузол стрижня (прив'язка (захоплений вузол) до вузла відображається в інформаційному полі рядка стану) і відпустіть кнопку миші;

- у вікні проекту з'явиться створений вами стрижень.

Видалення вузлів

Щоб видалити вузол натисніть кнопку **Видалити вузол**  на панелі побудови. Ви увійшли в режим видалення вузлів.

- перемістіть курсор миші у виділену область (прямокутна проектна область) у вікні проекту;

- перемістіть курсор на вузол, який ви хочете видалити (прив'язка (захоплений вузол) до вузла відображається в інформаційному полі рядка стану) і натисніть ліву або праву кнопку миші;


- вузол буде видалений.

Видалення стрижнів

Можливо кілька варіантів видалення стрижня:

Просте видалення стрижня


При простому видаленні стрижня, буде видалений стрижень що з'єднує два вузли, без видалення вузлів, котрі належать до більш ніж одного стрижня.

Щоб видалити стрижень натисніть кнопку **Видалити стрижень**  на панелі побудови. Ви увійшли в режим простого видалення стрижнів.

- перемістіть курсор миші у виділену область (прямокутна проектна область) у вікні проекту;
- перемістіть курсор на стрижень, який ви хочете видалити (прив'язка (захоплений стрижень) до стрижня відображається в інформаційному полі рядка стану);
- якщо обраний стрижень Вас влаштовує (він підсвітіться), то натисніть ліву або праву кнопку миші;
- стрижень буде видалений.

Інтелектуальне видалення стрижня


При інтелектуальному видаленні стрижня, буде видалений стрижень що з'єднує два вузли, з видаленням вузлів, котрі належать до більш ніж одного стрижня.

Щоб видалити стрижень натисніть кнопку **Видалити стрижень + вузли**  на панелі побудови. Ви увійшли в режим інтелектуального видалення стрижнів.

- перемістіть курсор миші у виділену область (прямокутна проектна область) у вікні проекту;
- перемістіть курсор на стрижень, який ви хочете видалити (прив'язка (захоплений стрижень) до стрижня відображається в інформаційному полі рядка стану);
- якщо обраний стрижень Вас влаштовує (він підсвітіться), то натисніть ліву чи праву кнопку миші;


- стержень буде видалений.

Завдання закріплень вузлів

Щоб задати закріплення вузлів, натисніть кнопку **Закріплення, навантаження, координати вузла**  на панелі побудови. Ви увійшли в режим завдання характеристик вузлів.

- перемістіть курсор миші у виділену область (прямокутна проектна область) у вікні проекту;
- перемістіть курсор на вузол, який ви хочете закріпити (прив'язка (захоплений вузол) до вузла відображається в інформаційному полі рядка стану) і натисніть ліву або праву кнопку миші;
- в меню оберіть пункт **Тип закріплення**;
- у вікні виберіть тип закріплення і натисніть кнопку **ОК** або клавішу **<Enter>**.


Завдання навантажень вузлів

Щоб задати сили вузлів, натисніть кнопку **Закріплення, навантаження, координати вузла**  на панелі побудови. Ви увійшли в режим завдання характеристик вузлів.

- перемістіть курсор миші у виділену область (прямокутна проектна область) у вікні проекту;
- перемістіть курсор на вузол, який ви хочете навантажити (прив'язка (захоплений вузол) до вузла відображається в інформаційному полі рядка стану) і натисніть ліву або праву кнопку миші;
- в меню оберіть пункт **Додані сили**;
- з'явиться вікно завдання сил;
- виберіть зі списку випадків навантаження необхідний вам випадок навантаження;
- задайте сили по X і Y для обраного випадку навантаження;
- якщо необхідно задати сили в цьому вузлі для іншого випадку навантаження, то натисніть кнопку **Змінити**, а потім виберіть наступний випадок навантаження і задайте сили, інакше натисніть кнопку **ОК**;

- якщо необхідно видалити один з випадків навантаження, виберіть його зі списку випадків навантажень, потім натисніть праву кнопку миші на рядку **Випадок навантаження** і виберіть з меню пункт **Видалити випадок навантаження**.

Коригування координат вузлів

Щоб змінити координати вузла / вузлів натисніть кнопку **Закріплення, навантаження, координати вузла**  на панелі побудови. Ви увійшли в режим завдання характеристик вузлів.

- перемістіть курсор миші у виділену область (прямокутна проектна область) у вікні проекту;
- перемістіть курсор на вузол, координати якого ви хочете змінити (прив'язка (захоплений вузол) до вузла відображається в інформаційному полі рядка стану) і натисніть ліву або праву кнопку миші;
 - в меню оберіть пункт **Координати вузла**;
 - з'явиться вікно зміни координат вузлів.


Зміна координат одного вузла

- задайте необхідні координати вузла і натисніть кнопку **ОК**;

Зміна координат групи вузлів


- в даному вікні можлива зміна координат будь-якого вузла / вузлів, для цього натискаючи кнопки **Попередній** і **Наступний** виберіть потрібний вузол, задайте необхідні координати і натисніть кнопку **Змінити** або клавішу **<Enter>**, повторіть ..., після завдання координат останнього змінюваного вузла, натисніть кнопку **ОК** або поспіль: клавішу **<Enter>** або кнопку **Змінити**, а потім кнопку **Скасувати**.

Зміна площі перерізу стрижнів

При створенні стрижня йому присвоюється за замовчуванням значення площі перетину рівне 0.1, якщо ж потрібно його підкоригувати, то натисніть кнопку **Площа перетину стрижня**  на панелі побудови. Ви увійшли в режим зміни площі перетину стрижнів.

- перемістіть курсор миші у виділену область (прямокутна проектна область) у вікні проекту;
- наведіть курсор на стрижень, площа перерізу якого ви хочете змінити (прив'язка (захоплений стрижень) до стрижня відображається в інформаційному полі рядка стану);
- якщо обраний стрижень Вас влаштовує (він підсвітіться) то натисніть ліву або праву кнопку миші;
- задайте в вікні площа перерізу стержня і натисніть кнопку **ОК**.

Використання підсистеми "Конструктор" для завдання ферменної конструкції

Щоб відобразити на екрані елемент системи "Конструктор" натисніть кнопку **Конструктор**  на панелі інструментів проекту.

За допомогою функцій елемента системи "Конструктор" можливо цифрове (текстове) створення ферми рис. 3.8.

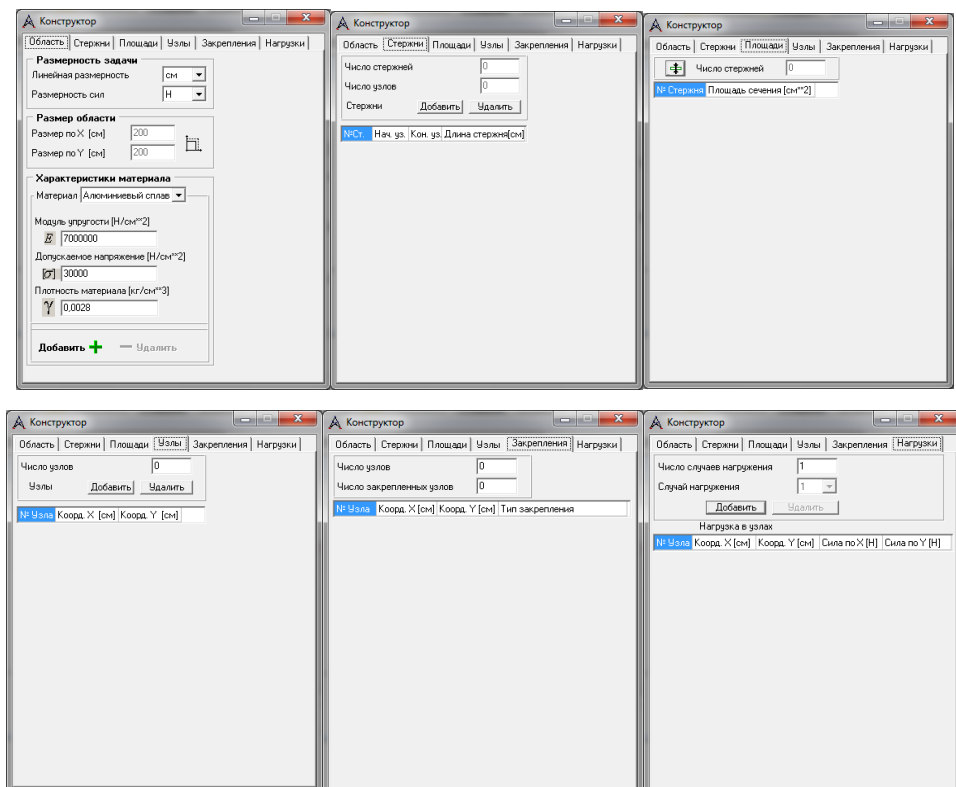


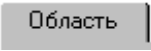

Рис. 3.8. Підсистеми "Конструктор" для завдання ферменної конструкції

Функції конструктора із створення ферми

- Зміна габаритів проектної області;

- Створення вузлів;
- Видалення вузлів;
- Створення стрижнів;
- Видалення стрижнів;
- Завдання закріплень вузлів;
- Завдання навантажень вузлів;
- Коригування координат вузлів;
- Зміна площі перерізу стержнів.

Зміна габаритів проєктної області

Щоб змінити габарити проєктної області виберіть закладку **Область**  і натисніть кнопку **Змінити розмір області**  в розділі **Розмір області**.

У вікні задайте розмір області по горизонталі **X**, потім за допомогою клавіші **<Tab>** на клавіатурі або за допомогою миші переведіть покажчик в другий осередок і задайте розмір області по вертикалі **Y**. Задавайте розміри, не порушуючи запропонованих в цьому вікні діапазонів зміни розмірів області. Ввівши необхідні розміри, натисніть кнопку **ОК** або клавішу **<Enter>**.

Доступні розміри габаритів проєктної області якщо:

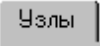
- **не заданий жоден елемент конструкції (новий проєкт)**

Доступні розміри габаритів проєктної області в системі від 0 до 9999.99 (включно) по **X** і **Y**, обраних лінійних одиниць (лінійної розмірності), з урахуванням формату дрібних чисел **x.XX** (два десяткових знаки після коми).

- **заданий будь-який елемент конструкції**

Доступні розміри габаритів проєктної області в системі від максимальних координат **X** і **Y** одного з вузлів до 9999.99 (включно) по **X** і **Y**, обраних лінійних одиниць (лінійної розмірності), з урахуванням формату дрібних чисел **x.XX** (два десяткових знаки після коми).

Створення вузлів

Щоб створити новий вузол / вузли виберіть закладку **Вузли** . Натисніть праву кнопку миші в будь-якому місці таблиці вузлів і виберіть пункт **Додати вузол [№ x]** з меню. З'явиться вікно створення нових вузлів.

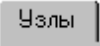
Створення одного нового вузла

Задайте координату **X** створюваного вузла, потім за допомогою клавіші **<Tab>** на клавіатурі або за допомогою миші переведіть покажчик в другий осередок і задайте координату **Y** створюваного вузла. Натисніть кнопку **ОК**.

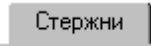
Створення групи нових вузлів

Задайте координату **X** створюваного вузла, потім за допомогою клавіші **<Tab>** на клавіатурі або за допомогою миші переведіть покажчик в другий осередок і задайте координату **Y** створюваного вузла. За допомогою клавіш **<Enter>** або кнопку **Додати**, повторіть вищенаведені операції для наступного вузла і т. д. Після завдання координат останнього вузла з групи натисніть кнопку **ОК** або поспіль: клавішу **<Enter>** або кнопку **Додати**, а потім кнопку **Скасувати**.

Видалення вузлів

Щоб видалити вузол виберіть закладку **Вузли** . Натисніть праву кнопку миші на рядку (таблиця вузлів) того вузла **x**, який ви хочете видалити і виберіть пункт **Видалити вузол [№ x]** з меню.

Створення стрижнів

Щоб створити новий стрижень / стрижні виберіть закладку **Стрижні** . Натисніть праву кнопку миші в будь-якому місці таблиці стрижнів і виберіть пункт **Додати стрижень [№ x]** з меню. З'явиться вікно створення нових стрижнів.

Створення одного нового стрижня

Задайте номер першого (початкового) вузла, потім за допомогою клавіші **<Tab>** на клавіатурі або за допомогою миші переведіть покажчик в другий осередок і задайте номер другого (кінцевого) вузла створюваного стрижня. Натисніть кнопку **ОК**.

Створення групи нових стрижнів

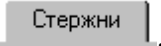
Задайте номер першого (початкового) вузла, потім за допомогою клавіші <Tab> на клавіатурі або за допомогою миші переведіть покажчик в другий осередок і задайте номер другого (кінцевого) вузла створюваного стрижня. За допомогою клавіш <Enter> або кнопку **Додати**, повторіть вищенаведені операції для наступного стержня і т.д. Після створення номера початкового і кінцевого вузла останнього створюваного стержня з групи натисніть кнопку **ОК** або поспіль: клавішу <Enter> або кнопку **Додати**, а потім кнопку **Скасувати**.

Видалення стрижнів

Можливо кілька варіантів видалення стержня:

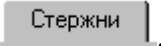
Просте видалення стержня

При простому видаленні стрижня, буде видалений стрижень що з'єднує два вузли, без видалення вузлів, котрі належать до більш ніж одного стрижня.

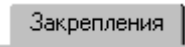
Щоб видалити стрижень виберіть закладку **Стрижні** . Натисніть праву кнопку миші на рядку (таблиця стрижнів) того стержня х, який ви хочете видалити і виберіть пункт **Видалити стрижень [№ х]** з меню.

Інтелектуальне видалення стержня

При інтелектуальному видаленні стрижня, буде видалений стрижень що з'єднує два вузли, з видаленням вузлів, котрі належать до більш ніж одного стрижня.

Щоб видалити стрижень виберіть закладку **Стрижні** . Натисніть праву кнопку миші на рядку (таблиця стрижнів) того стержня х, який ви хочете видалити і виберіть пункт **Видалити стрижень [№ х] + вузли** з меню.

Завдання закріплень вузлів

Щоб встановити розмір закріплення вузла виберіть закладку **Закріплення** .

Натисніть ліву кнопку миші на рядку (таблиця закріплень) того вузла, який ви хочете закріпити (змінити тип закріплення) в стовпці **Тип закріплення**. Виберіть із запропонованого списку тип закріплення обраного вузла.

Завдання навантажень вузлів

Щоб задати навантаження вузла виберіть закладку навантажений

- виберіть зі списку випадків навантаження необхідний вам випадок навантаження;

Завдання навантажень вузлів

Щоб задати навантаження вузла виберіть закладку **Навантаження**

Нагружения

- виберіть зі списку випадків навантаження необхідний вам випадок навантаження;

- задайте сили по X і Y для обраного випадку навантаження:

- виберіть осередок, що відповідає номеру навантажувати вузла і додається силі по X або Y (стовпці **Сила по X** або **Сила по Y**), натиснувши на ній лівою кнопкою миші, вона підсвітиться пунктирним обідком;

- увійдіть в режим редагування, натиснувши клавішу **<Enter>** або ліву кнопку миші;

- задайте значення сили і натисніть клавішу **<Enter>**;

- якщо необхідно задати сили в цьому вузлі для іншого випадку навантаження, то виберіть наступний випадок навантаження і задайте сили;

- якщо в списку випадків навантаження немає потрібного вам то створіть його, натиснувши праву кнопку миші в будь-якому місці таблиці навантажень або на рядку **Випадок навантаження** і виберіть з меню пункт **Додати випадок навантаження**;

- якщо необхідно видалити один з випадків навантаження, виберіть його зі списку випадків навантажень, потім натисніть праву кнопку миші в будь-якому місці таблиці навантажень або на рядку **Випадок навантаження** і виберіть з меню пункт **Видалити випадок навантаження**;

Коригування координат вузлів

Щоб змінити координати вузла / вузлів виберіть закладку **Вузли**

Узлы 1

Зміна однієї з координат вузла

- виберіть осередок, що відповідає номеру вузла і змінною координаті X або Y (стовпці **Коорд. X** або **Коорд. Y**), натиснувши на ній лівою кнопкою миші, вона підсвітиться пунктирним обідком;

- увійдіть в режим редагування, натиснувши клавішу **<Enter>** або ліву кнопку миші;

- задайте значення координати і натисніть клавішу **<Enter>**;

Зміна координат вузла / вузлів

- натисніть праву кнопку миші на рядку (таблиця вузлів) того вузла x, координати якого ви хочете змінити і виберіть пункт **Координати вузла з меню**;

- з'явиться вікно зміни координат вузлів:

Зміна координат одного вузла

- задайте необхідні координати вузла і натисніть кнопку **ОК**.

Зміна координат групи вузлів

- в даному вікні можлива зміна координат будь-якого вузла / вузлів, для цього натискаючи кнопки **Попередній** і **Наступний** виберіть потрібний вузол, задайте необхідні координати і натисніть кнопку **Змінити** або клавішу **<Enter>**, повторіть після завдання координат останнього змінюваного вузла, натисніть кнопку **ОК** або поспіль: клавішу **<Enter>** або кнопку **Змінити**, а потім кнопку **Скасувати**.

Зміна площі перерізу стержнів

При створенні стрижня йому присвоюється за замовчуванням значення площі перетину рівне **0.1**, якщо ж потрібно його підкоригувати, то виберіть закладку **Площі**


Площади

- виберіть осередок, що відповідає номеру стрижня площа перерізу якого ви хочете змінити (стовпець **Площа перетину**), натиснувши на ній лівою кнопкою миші, вона підсвітиться пунктирним обідком;

- увійдіть в режим редагування, натиснувши клавішу <Enter> або ліву кнопку миші;

- задайте значення площі перетину стрижня і натисніть клавішу <Enter>.


3.3.4. Розрахунок на міцність ферменної конструкції

Щоб зробити простий розрахунок ферменної конструкції натисніть кнопку **Розрахунок на міцність** . Після закінчення розрахунку з'явиться інформаційне вікно, яке повідомить, що розрахунок закінчений або перерваний по тій або іншій причині. Щоб продовжити роботу, натисніть в ньому кнопку **ОК**.

Результати розрахунку на міцність

Натисніть кнопку **Результати розрахунку на міцність** .

При перегляді результатів простого розрахунку, в вікні (формі) результатів простого розрахунку ми можемо вибрати дані для аналізу **Переміщення вузлів** або **Напруга в стрижнях**, натиснувши лівою кнопкою миші на відповідних рядках. Також ми можемо вибрати випадок навантаження зі списку випадків навантаження, натиснувши лівою кнопкою миші на цьому списку і вибравши потрібний нам випадок навантаження.

Для продовження роботи в системі натисніть кнопку **ОК** або кнопку **Закрити**  в системному меню вікна результатів простого розрахунку.

Графічне представлення результатів розрахунку на міцність


Для перегляду результатів простого розрахунку в графічному вигляді натисніть кнопку **Графічне представлення результатів розрахунку** , після чого з'явиться форма (вікно) графічного представлення результатів простого розрахунку з двома видами відображення результатів, показана на рис. 3.9.-3.10.



Рис.3.9. Напряжено-деформованний стан ферменної конструкції

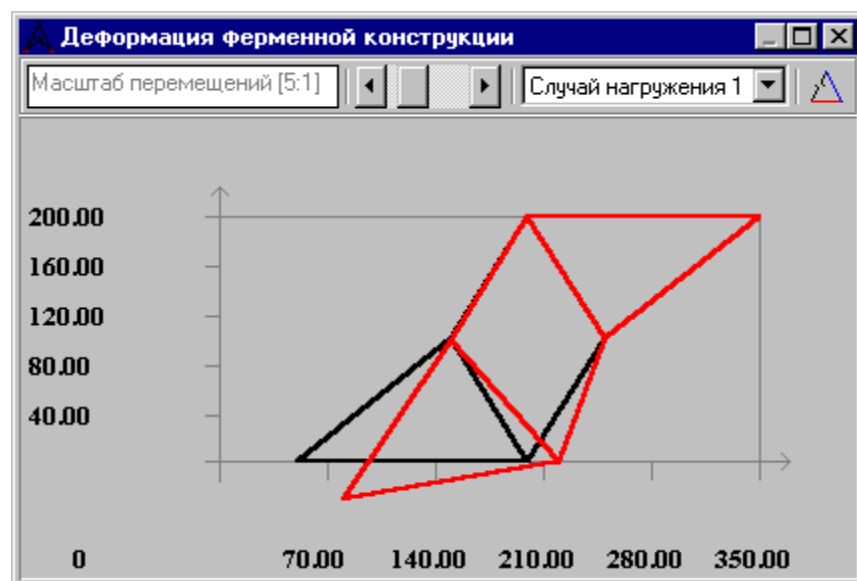



Рис. 3.10. Деформованний стан ферменної конструкції

Для продовження роботи в системі натисніть кнопку **Закрити**  в системному меню вікна графічного представлення результатів простого розрахунку.

3.4. Приклад розробки силової схеми ферменної конструкції

Вважаються заданими навантаження, умови закріплення, а також габарити проектної області, в якій необхідно розмістити і спроектувати плоску ферменну конструкцію.

Необхідно знайти раціональну силову схему плоскої ферменної конструкції. Конкретне завдання наведено на рис. 3.11. У заданій прямокутній плоскій області потрібно розмістити найбільш раціональну силову схему ферменної конструкції.

Тут $P_I = P_{II} = P_{III} = 10000\text{Н}$ - три випадки навантаження. Матеріал - алюміній.

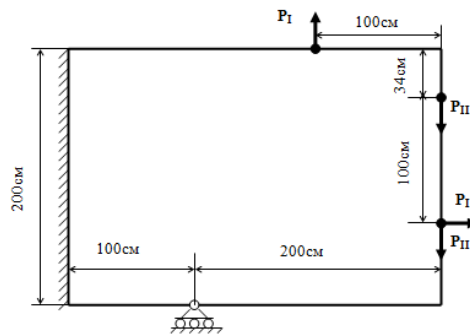


Рис. 3.11. Умова завдання

Вважається що:

- а) сили P_I , P_{II} і P_{III} діють не одночасно (три розрахункових випадку);
- б) тип закріплення по лівій межі позначає, що ферму можна закріпити в будь-якій точці цієї межі і число цих точок необмежено. При цьому тип закріплення довільний.
- в) нижня опора означає, що в цій точці ферма може бути закріплена тільки по осі Y ;
- г) проєктувальник може використовувати, а може і не використовувати нижню опору. Те ж саме відноситься і до закріплення по лівій межі області.

Приклад формування в підсистемі «ферменна конструкція» силової схеми, що відповідає завданням рис. 3.12 (тут $P_I = P_{II} = P_{III} = 10000\text{Н}$ - три розрахункових випадку, матеріал - алюміній, $E = 70000\text{Н/см}^2$, $[\sigma] = 30000\text{Н/см}^2$), наведено на рис. 3.14 (показаний тільки один розрахунковий випадок - перший). Числові значення вихідних даних в тому вигляді, якому вони задаються в системі, наведені на рис 3.15.

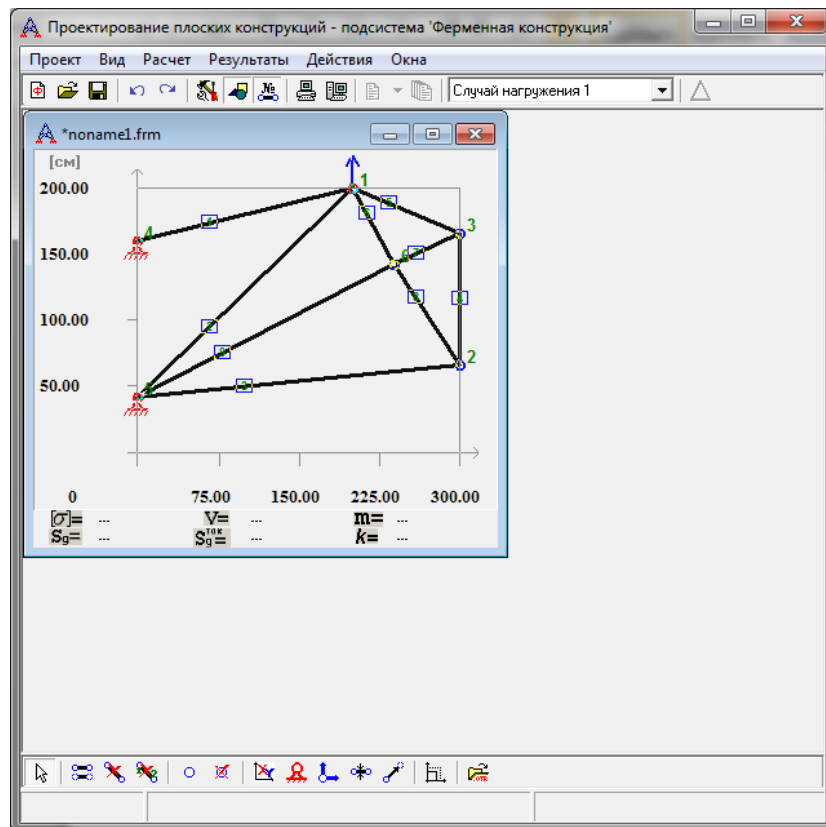


Рис. 3.12. Формування в підсистемі «ферменна конструкція» силова схема.

Конструктор

Область | Стержни | Площади | Узлы | Закрепления | Нагрузки

Размерность задачи: Линейная размерность: Размерность сил:

Размер области: Размер по X [см]: Размер по Y [см]:

Характеристики материала: Материал: Модуль упругости [Н/см²]: Допускаемое напряжение [Н/см²]: Плотность материала [кг/см³]:

Добавить + Удалить -

Конструктор

Область | Стержни | Площади | Узлы | Закрепления | Нагрузки

Число стержней: Число узлов:

Стержни:

№Ст.	Нач. уз.	Кон. уз.	Длина стержня[см]
1	1	4	203.96
2	1	5	254.88
3	5	2	300.96
4	2	3	100
5	3	1	105.62
6	1	6	69.89
7	3	6	65.55
8	6	2	97.45
9	5	6	259.08

Конструктор

Область | Стержни | Площади | Узлы | Закрепления | Нагрузки

Число стержней:

№ Стержня	Площадь сечения [см ²]
1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1

Конструктор

Область | Стержни | Площади | Узлы | Закрепления | Нагрузки

Число узлов: Узлы:

№ Узла	Коорд. X [см]	Коорд. Y [см]
1	200	200
2	300	66
3	300	166
4	0	160
5	0	42
6	239	142

Конструктор

Область | Стержни | Площади | Узлы | Закрепления | Нагрузки

Число узлов: Число закрепленных узлов:

№ Узла	Коорд. X [см]	Коорд. Y [см]	Тип закрепления
1	200	200	Не закреплен
2	300	66	Не закреплен
3	300	166	Не закреплен
4	0	160	Закреплен по X и Y
5	0	42	Закреплен по X и Y
6	239	142	Не закреплен

Конструктор

Область | Стержни | Площади | Узлы | Закрепления | Нагрузки

Число случаев нагружения: Случай нагружения:

Нагрузка в узлах:

№ Узла	Коорд. X [см]	Коорд. Y [см]	Сила по X [Н]	Сила по Y [Н]
1	200	200	0	10000
2	300	66	0	0
3	300	166	0	0
4	0	160	0	0
5	0	42	0	0
6	239	142	0	0

Рис. 3.13. Завдання умови задачі

У процесі роботи над проєктом практично на кожному етапі проводиться розрахунок конструкції методом кінцевих елементів, в результаті чого в кожному стрижні визначаються зусилля N_i , напруги $\sigma_i = N_i / F_i$, коефіцієнт запасу міцності (або стійкості), а для стислих стрижнів - необхідні з умов стійкості значення мінімальних моментів інерції. Крім чисто міцнісного розрахунку у всіх випадках автоматично підраховується силова вага конструкції S_G . Частина цих додаткових характеристик виводяться на екран (рис.3.12) під зображенням ферми.

Для того щоб стрижень не зруйнувався, напруги σ_i повинні бути менше допустимих $[\sigma]$, а коефіцієнт запасу міцності повинен бути більшим за одиницю. Але це справедливо тільки для розтягнутих стрижнів. Для стиснутих стрижнів все дещо ускладнюється. Очевидно, стислий стрижень може втратити стійкість задовго до того, як буде досягнуто граничне значення допустимої напруги $[\sigma]$ для даного матеріалу. Тому, якщо для розтягнутих стрижнів умова міцності записується як $\sigma_i \leq [\sigma]$, то для стислих стрижнів ця

умова міцності буде $\sigma_i \leq \sigma_{кр}$, де $\sigma_{кр} = P_{кр} / F_i$. У свою чергу, значення критичної

сили втрати стійкості для стисненого стержня визначається формулою Ейлера:

$$P_i^{кр} = \pi^2 E J_i / \ell_i^2, \text{ де } J_i - \text{мінімальний момент інерції поперечного перерізу } i\text{-го}$$

стержня, E - модуль пружності матеріалу, ℓ_i - довжина i -го стержня. Як бачимо, для стислих стрижнів величезне значення має форма поперечного перерізу, так як від форми залежить момент інерції перерізу.

Тому стислі стрижні доводиться робити не тільки з більшою площею поперечного перерізу, але і з цілком певним значенням моменту інерції цього перерізу. Облік явища втрати стійкості - необхідна складова розрахунку більшості конструкцій. В системі проєктування плоских конструкцій реалізована інженерна методика визначення необхідних площ поперечних перерізів стиснутих стрижнів і моментів інерції цих перетинів. В основу цієї методики покладено зменшення значення допустимої напруги для стиснутих

стержнів, а саме $[\sigma]_{сж} = \phi[\sigma]$, де ϕ - коефіцієнт поздовжнього вигину або коефіцієнт зниження основної допустимої напруги.

Коефіцієнт ϕ визначається за спеціальними таблицями залежно від прийнятого розробником коефіцієнта λ гнучкості стиснутих стрижнів. Коефіцієнт λ надалі завжди повинен бути більше коефіцієнта мінімально-можливої гнучкості $\lambda_{пред} = \pi \sqrt{\frac{E}{[\sigma]}}$ (для більшості матеріалів його можна взяти рівним 50). Після отримання ϕ в кожному розрахунковому випадку визначається необхідна площа поперечного перерізу стиснутого стрижня $F_i^{сж} = N_i^{сж} / [\sigma]_{сж}$, потім мінімальний радіус інерції $i_{мин} = l / \lambda$ і мінімально допустиме значення моменту інерції для даного стрижня $I_i = i_{мин}^2 F_i^{сж}$. Тут $N_i^{сж}$ - стискає зусилля в стрижні. Коефіцієнт запасу стійкості визначається за формулою $K_i = [\sigma]_{сж} / \sigma_i^{сж}$, де $\sigma_i^{сж} = N_i^{сж} / F_i$.

Варто тільки пам'ятати, що збільшення λ призводить до зменшення ϕ , а це, в свою чергу, до збільшення $F_i^{сж}$, що негативно позначається на зменшенні маси конструкції. Але найчастіше на це доводиться йти з метою зменшення потрібного моменту інерції для даного стрижня.

Необхідна площа стрижня з умови міцності на розтяг в кожному розрахунковому випадку визначається за формулою $F_i^{раст} = N_i^{раст} / [\sigma]$, де $N_i^{раст}$ - розтяжне зусилля в стрижні, а коефіцієнт запасу міцності на розтягнення - за формулою $K_i = [\sigma] / \sigma_i^{раст}$, де $\sigma_i^{раст} = N_i^{раст} / F_i$.

На рис. 3.16 показано, що використання цієї методики розрахунку на стійкість стиснутих стрижнів при заданому коефіцієнті гнучкості $\lambda=50$ привело до зменшення значень допустимих напружень стиснення в порівнянні з допустимими напруженнями розтягу. Тобто по суті, для стислих стрижнів визначальними стають не допустимі напруження розтягування, а допустимі напруження стиснення, що призводить до зменшення коефіцієнтів запасу

стійкості і збільшення потрібних площ. При цьому в таблиці додатково приведені і мінімально необхідні моменти інерції стислих стрижнів.

На рис 3.14 наведена таблиця з результатами розрахунку на міцність для одного розрахункового випадку для конструкції. На малюнку 3.17 представлена графічна інтерпретація результатів розрахунку для 1-го випадку навантаження, де в деякому масштабі показані переміщення вузлів і позначені стрижні, які не витримують навантаження.

Як відомо, основним результатом міцнісного розрахунку є значення переміщень вузлів ферми і напруг в стрижнях, що утворюють ферму. Розрахунок проводиться при заданих значеннях площ F_i поперечних перерізів стрижнів і заданих характеристиках матеріалу, з якого виготовлена конструкція. В даному випадку площа поперечних перерізів для всіх стрижнів прийнята однаковою і рівною $F_i = 0,1 \text{ см}^2$, а в якості матеріалу взято алюміній з допускаються напругою $[\sigma]=30000 \text{ Н/см}^2$.

Напряжения в стрижнях													
Заданные площади			Напряжения Н/см2			Коефф. запаса прочности			Потребные площади [см2]			Необх. моменты инерции [см4]	
№ стерж.	см2	Случай 1	Случай 2	Случай 3	Случай 1	Случай 2	Случай 3	Случай 1	Случай 2	Случай 3	Случай 1	Случай 2	Случай 3
1	1.000e-01	1.73e+05	0.00e+00	0.00e+00	1.17e-01	#####	#####	8.52e-01	0.00e+00	0.00e+00	1.42e+01	#####	#####
2	1.000e-01	1.91e+05	0.00e+00	0.00e+00	1.57e-01	#####	#####	6.37e-01	0.00e+00	0.00e+00	#####	#####	#####
3	1.000e-01	2.14e+04	0.00e+00	0.00e+00	9.49e-01	#####	#####	1.06e-01	0.00e+00	0.00e+00	3.92e+00	#####	#####
4	1.000e-01	2.93e+04	0.00e+00	0.00e+00	7.17e-01	#####	#####	1.39e-01	0.00e+00	0.00e+00	5.58e-01	#####	#####
5	1.000e-01	4.07e+04	0.00e+00	0.00e+00	4.99e-01	#####	#####	2.01e-01	0.00e+00	0.00e+00	8.96e-01	#####	#####
6	1.000e-01	3.43e+04	0.00e+00	0.00e+00	8.75e-01	#####	#####	1.14e-01	0.00e+00	0.00e+00	#####	#####	#####
7	1.000e-01	4.14e+04	0.00e+00	0.00e+00	7.24e-01	#####	#####	1.38e-01	0.00e+00	0.00e+00	#####	#####	#####
8	1.000e-01	3.41e+04	0.00e+00	0.00e+00	8.80e-01	#####	#####	1.14e-01	0.00e+00	0.00e+00	#####	#####	#####
9	1.000e-01	4.42e+04	0.00e+00	0.00e+00	6.79e-01	#####	#####	1.47e-01	0.00e+00	0.00e+00	#####	#####	#####

Перемещения узлов									
Коорд. узлов [см]		Перемещения по X [см]			Перемещения по Y [см]				
№ узла	X	Y	Случай 1	Случай 2	Случай 3	Случай 1	Случай 2	Случай 3	
1	2.00e+02	2.00e+02	9.89e+00	0.00e+00	0.00e+00	2.37e+01	0.00e+00	0.00e+00	
2	3.00e+02	6.60e+01	3.16e+00	0.00e+00	0.00e+00	2.79e+01	0.00e+00	0.00e+00	
3	3.00e+02	1.66e+02	9.25e+00	0.00e+00	0.00e+00	2.75e+01	0.00e+00	0.00e+00	
4	0.00e+00	1.60e+02	2.42e+09	0.00e+00	0.00e+00	4.84e+10	0.00e+00	0.00e+00	
5	0.00e+00	4.20e+01	2.42e+09	0.00e+00	0.00e+00	1.91e+09	0.00e+00	0.00e+00	
6	2.39e+02	1.42e+02	8.41e+00	0.00e+00	0.00e+00	2.43e+01	0.00e+00	0.00e+00	

Доп. напр. растяжения [Н/см ²]	$[\sigma] =$	3.000e+04
Доп. напр. сжатия [Н/см ²]	$[\sigma]_{\text{сж}} =$	2.028e+04
Силовой вес конструкции [Н*см]	$S_0 =$	1.175e+07
Силовой вес ТОК [Н*см]	$S_0^{\text{ТОК}} =$	Нет данных
Вес конструкции / Вес ТОК	$k =$	Нет данных
Объем материала [см ³]	$V =$	1.457e+02
Масса конструкции [кг]	$m =$	4.081e-01

Дополнительные параметры расчета	
λ	50
<input checked="" type="checkbox"/>	Отображать цветовую индикацию
<input type="checkbox"/>	Перемещения в процентах
<div>Пересчитать</div> <div>Закрыть</div>	

Рис. 3.14. Таблица з результатами розрахунку на міцність для одного розрахункового випадку для конструкції

Перш за все, слід звернути увагу на небезпечні величини коефіцієнтів запасу міцності (стійкості), значення яких для всіх стрижнів у всіх

розрахункових випадках менше одиниці. Це говорить про те, що площі стрижнів, за замовчуванням задані рівними 0,1 см², недостатні для того, щоб стрижні сприймали навантаження без руйнування (тобто напруги в них перевищують допустиме напруження на розтяг або критичне напруження при стисканні). Зрозуміло також, що хоча б з цієї причини ця конструкція непрацездатна і потрібно її модифікація шляхом зміни значень площ поперечного перерізу стержнів.

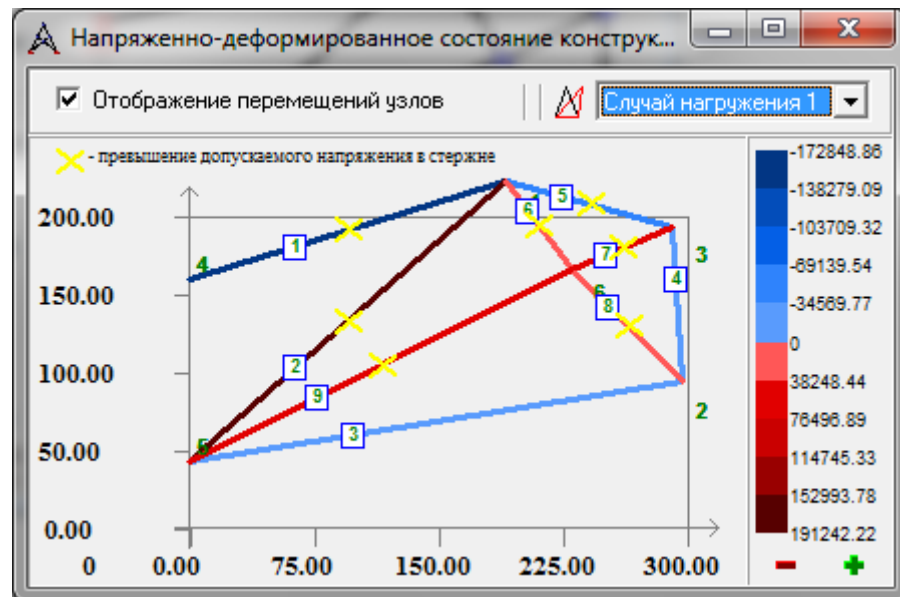


Рис.3.15. Графічна інтерпретація результатів розрахунку для 1-го випадку навантаження

3.5. Висновки до розділу

У цьому розділі були розроблені загальний алгоритм роботи програми і інтерфейс на основі передбачуваної функціональності, були обрані програмні засоби для реалізації поставленого завдання. Програма реалізована на РС для роботи під управлінням ОС Microsoft® Windows 10 в системі програмування для Windows - Borland Delphi 7 і Visual Fortran.

Система проектування будівельних конструкцій призначена для розрахунку і оптимізації плоских ферменних конструкцій. За допомогою системи можна вирішувати такі завдання:

- Проектування силової схеми ферменних конструкцій.
- Розрахунок ферми.

- Оптимізація силової схеми ферменних конструкцій.

ВИСНОВКИ

У магістерській роботі було розглянуто розробку програми для проєктування і розрахунку будівельних конструкцій.

Для досягнення поставленої мети були вирішені наступні завдання:

- Розглянуто основні завдання які вирішуються в САПР, наведена узагальнена схема процесу автоматизованого проєктування. Розглянуто типову структуру САПР, яка базується на звичайно елементному аналізі.

- Проведено аналіз обчислювальних комплексів на основі МСЕ. Відзначено, що сучасні програмні комплекси автоматизації проєктування можна умовно розділити на три підсистеми: препроцесор, процесор і постпроцесор. Препроцесор відповідає за підготовку вхідних даних, яка включає такі етапи роботи, як опис геометричної моделі проєктованого об'єкта, і його дискретизацію на заданий тип кінцевих елементів. Процесор виконує всі необхідні для конкретного типу завдання розрахунки: формує матриці мас, жорсткостей; будує систему лінійних алгебраїчних рівнянь; враховує початкові і граничні умови; вирішує систему рівнянь і виводить результати розрахунку. Постпроцесор автоматизує процес аналізу результатів і генерацію документації. Підкреслено, що в залежності від можливостей ядра таких САПР - процесора, їх можна умовно розділити на два великі класи: універсальні - призначені для автоматизації аналізу великої кількості типів завдань механіки, і спеціалізовані програмні комплекси - орієнтовані на розрахунок конструкцій спеціального типу, наприклад, тільки пластин і ферм. На перший погляд, універсальні програмні комплекси переважають, оскільки вони дозволяють автоматизувати процес проєктування великої кількості типів об'єктів. Однак така універсальність, як правило, призводить до громіздкості програмного комплексу, який часто робить його використання складним і незручним. Крім того, в процесі розробки таких систем на користь універсальності часто жертвують точністю розрахунків. Тому спеціалізовані САПР, орієнтовані на рішення вузьких класів задач, на сьогодні залишаються дуже популярними і постійно розвиваються. Вони зазвичай мають такі

переваги як гнучкість, ефективність, а також зручність і простота в експлуатації. Крім того, при розробці таких систем, як правило, вдається домогтися високої точності результатів розрахунків засобами вживання ефективних обчислювальних схем, орієнтованих на вузькі класи завдань або методів.

- Розглянуто проектування силових конструкцій на прикладі ферменної конструкції. Розглядаються евристичний підхід і підхід на основі методу силового аналізу. В роботі присутній опис методики проектування силових схем ферменних конструкцій методом силового аналізу.

- Розроблено та реалізовано систему проектування і розрахунку будівельних конструкцій. Система реалізована на РС для роботи під управлінням ОС Microsoft® Windows 10 в системі програмування для Windows - Borland Delphi 7 і Visual Fortran. За допомогою системи можна вирішувати такі завдання:

Проектування силової схеми.

Заданими вважаються навантаження, умови закріплення та межі області прямокутної форми, в яку повинна бути вписана плоска ферма. Потрібно знайти раціональну за критерієм маси силову схему (структуру) ферменної конструкції.

Розрахунок ферми.

Заданими вважаються силова схема і розподіл матеріалу в ферменній конструкції, навантаження, умови закріплення. Потрібно визначити переміщення вузлів, напруження і зусилля в стержнях ферми, а також необхідні з умов міцності і стійкості площі поперечного перерізу стержнів і мінімальні моменти інерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3321_2003 Система конструкторської документації. Терміни та визначення основних понять. – [Чинний від 2003-12-08]. Вид. офіц. Київ: Держстандарт України, 2005. 51 с.
2. ДСТУ 2226-93. Автоматизовані системи. Терміни та визначення. – [Чинний від 1994-07-01]. Вид. офіц. Київ : Держстандарт України, 1994. 93 с. http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=61937
3. ДСТУ ISO/IEC/IEEE 15288:2016 (ISO/IEC/IEEE 15288:2015, IDT) Інженерія систем і програмного забезпечення. Процеси життєвого циклу систем. [На заміну ДСТУ ISO/IEC/IEEE 15288:2015; чинний від 2018-01-01]. Вид. офіц. Київ: ДП «УкрНДНЦ», 2018. 90 с
4. Безменов М. І. Основи програмування у середовищі Delphi : навч. посіб. – Харків : НТУ «ХП», 2010. – 608 с.
5. Борисенко В. Д. Основи комп'ютерного моделювання в інженерній діяльності: навчальний посібник / В. Д. Борисенко, С. А. Устенко, І. В. Устенко. – Миколаїв: МНУ, 2016. – 276 с.
6. Гервас О.Г. САПР об'єктів середовища. Навчально-методичний посібник / О.Г. Гервас. – Умань: Візаві, 2018. - 160 с.
7. Гудима Ю.В. Системи автоматизованого проектування технологічних процесів: Конспект лекцій для студентів заочної форми навчання. – Чернівці: Рута, 2003. – 44 с.
8. Князь, І. О. Комп'ютерне моделювання динамічних систем. Розділ "Основи комп'ютерного моделювання" : навчальний посібник [Текст] / І. О. Князь. – Суми : Сумський державний університет, 2011. – 102 с
9. Наумчук О. М. Основи систем автоматизованого проектування: Інтерактивний комплекс навчально-методичного забезпечення./ О.М. Наумчук. – Рівне: НУВГП, 2008.-136с.

10. Павленко П. М. Автоматизовані системи технологічної підготовки розширених виробництв. Методи побудови та управління: Монографія. – К.: Книжкове вид-во НАУ, 2005. – 280 с.
11. Програмне забезпечення інженерних розрахунків : конспект лекцій для студентів спеціальності 192 «Будівництво та цивільна інженерія» всіх форм навчання / Укладач : Сорочак А.П. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2018. – 128 с.
12. Розв'язок задач проєктування приладів та систем з використанням ANSYS і MATHCAD : підручник / І. А. Гришанова, Л. П. Згуровська, Ю. В. Киричук. – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2022. – 180 с.
Системи автоматизованого проєктування. URL: <https://msd.com.ua/osnovy-proektirovaniya-ximicheskix-proizvodstv-ioborudovaniya/sistemy-avtomatizirovannogo-proektirovaniya>
13. Стенін О. А., Лапковський С. В., Солдатова М. О. Використання CALS-технологій в сучасній промисловості // Адаптивні системи автоматичного управління : міжвідомчий науково-технічний збірник. 2011. № 18(38). С. 114–123. URL: <https://ela.kpi.ua/handle/123456789/4934>
14. Саєнко С. Ю. Основи САПР / С. Ю. Саєнко, І. В. Нечипоренко – Х. : ХДУХТ, 2017.
15. Вислоух, С. П. Інформаційні технології в задачах технологічної підготовки приладо- та машинобудівного виробництва [Електронний ресурс]: монографія / С. П. Вислоух ; НТУУ «КПІ». – Електронні текстові данні (1 файл: 15,47 Мбайт). – Київ : НТУУ «КПІ», 2011. – 482 с. – URL: <https://ela.kpi.ua/handle/123456789/32767>
16. Системи автоматизованого проєктування. Конспект лекцій [Електронний ресурс]: навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою «Комп'ютерно-інтегровані

- технології виробництва приладів» спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад. К. С. Барандич, О. О. Подолян, М. М. Гладський. – Електронні текстові дані (1 файл 3,13 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 97 с. – [URL:https://ela.kpi.ua/handle/123456789/45614](https://ela.kpi.ua/handle/123456789/45614)
17. S. P. Vysloukh, V. S. Antonyuk, K. S. Barandych, O. V. Voloshko, “Mathematical modeling of automated production systems”, International Scientific Journals Mathematical Modeling, vol. 4, Is. 4, pp. 114-120, 2020
 18. Catherine Barandych, Sergey Vyslouh, Victor Antoniuk, Oleksandr Tymoshenko, Viktor Koval. Lathe Turning Mode Optimization for Parts Working under Conditions of Cyclic Loading. Ukrainian Journal of Mechanical Engineering and Materials Science. 2016. Vol. 2, No. 2. pp. 53– 60
 19. Ситник, В. Ф. Імітаційне моделювання: Навч.-метод. Посібник для самост. вивч. дисц [Текст] / В. Ф. Ситник, Н. С. Орленко. – К. : КНЕУ, 1999. – 208 с.
 20. Соколова Н.О. Програмування в середовищі DELPHI. Практичне видання./ Н.О. Соколова. – Темплан, 2009. – 36 с.
 21. А. В. Петров, В. М. Черненко, Разработка САПР. Проблемы и принципы создания САПР. книга1, М., "Высшая школа", 1990
 22. Абрамов Н.Н., Беркун В.Б., Кучеренко В.В., Перекальский В.М. Эффективные итерационные алгоритмы решения тепловых задач: Учебное пособие – М.: МИСИ, 1987. 67 с.
 23. Автоматизация проектирования авиационных конструкций, Межвузовский сборник, Куйбышев, 1979
 24. Архангельский А.Я. Программирование в Delphi 6. – М.: «Издательство БИНОМ », 2002г. – 1120 с.
 25. Басов К.А. ANSYS в примерах и задачах. М.: КомпьютерПресс, 2002.

26. Боровков А.И. Программный комплекс конечно-элементного анализа
FEA // Аннотированный каталог учебных программных средств.
Вып.3. СПб: СПбГТУ, 1995. С.100-102.
27. Бурман Я.З., Салахов Р.Р. О реализации МКЭ на персональных ЭВМ.
Прикладные проблемы информатики, No1, 1989.
28. Бурышкин М.Л., Гордеев В.Н. Эффективные методы и программы
расчета на ЭВМ симметричных конструкций. Киев: Будивельник,
1984.120 с.
29. Быков В.П. Методическое обеспечение САПР в машиностроении
/Быков В.П. – Л.: Машиностроение, 1989. – 255с.
30. В. Е. Климов, Разработка САПР. Графические системы САПР.
книга7, Москва, "Высшая школа", 1990
31. Вилипыльд Ю.К., Лайгна К.Ю., Кала Т.Н. Расчет стержневых и
пластинчатых систем по методу конечных элементов МКЭ/20.
Таллинн, 1979.
32. Г. Шпур, Ф.-Л. Краузе, Автоматизированное проектирование в
машиностроении. М., "Машиностроение", 1988
33. Галкин Д.С., Галкина Н.С., Гусак Ю.В. Многоцелевая
автоматизированная расчетная система МАРС. Сб.: Комплексы
программ математической физики. – Новосибирск, 1984.
34. Гоменюк С.И. Объектно-ориентированные модели и методы анализа
механических процессов /Гоменюк С.И. – Никополь: Никопольская
коммунальная типография, 2004. – 316 с.
35. Городецкий А.С. Программа МИРАЖ для статического расчета
конструкций методом конечных элементов. Автоматизация
проектирования как комплексная проблема совершенствования
проектного дела в стране: Сб. трудов всесоюзной научной
конференции. М., 1973.
36. Городецкий А.С., Здоренко В.С. Типовая проектирующая подсистема
ЛИРА для автоматизированного проектирования несущих

- строительных конструкций. Сб.: Системы автоматизированного проектирования объектов строительства. Вып.1, 1982.
37. Гофман В.Э. Delphi 5. В подлиннике, СПб., 2000
38. Зенкевич О. Метод конечных элементов в в технике. М.: Мир, 1975. 536 с.
39. Колодницкий Н.М., Левицкий В.Г. Обзор основных программных средств для моделирования математических задач // САПР и графика, 1999. - № 10. - С. 56-65.
40. Комаров В. А., Соловов А. В., Черепашков А.А. Проектирование ферменных конструкций в режиме диалога с ЭВМ, Куйбышев, КуАИ, 1985.
41. Комаров В.А. О рациональных силовых конструкциях крыльев малого удлинения. – В кн. Проектирование оптимальных конструкций. – Куйбышев, КУАИ, 1968, вып. 32, с. 6-26.
42. Ли К. Основы САПР (CAD/CAM/CAE) /Ли К. – СПб.: Питер, 2004. – 560с.
43. Молчанов И.Н., Николенко Л.Д. Основы метода конечных элементов. Киев: Наукова Думка, 1989. 272 с.
44. Никольский М.Д., Чернева И.М., Безперстова Н.Ф.и др. Комплекс программ MORE для расчета сооружений по методу конечных элементов. В книге: Экспериментальные и теоретические исследования по механике твердого деформируемого тела. Сб. трудов ЛИИЖТ. – Л.: ЛИИЖТ, 1978.
45. Новые направления оптимизации в строительном проектировании. Под ред. Э. Атрена, Р.Г. Галлагера и др., М., Стройиздат, 1989.
46. Норенков И.П. Основы автоматизированного проектирования /Норенков И.П. - М.: МГТУ им. Н. Э. Баумана, 2002. - 334 с.
47. Норенков И. П. Разработка систем автоматизированного проектирования /Норенков И. П. – М.: Изд-во МГТУ им. Н.Э.Баумана, 1994. – 207с.

48. Постнов В.А. Проблемы автоматизации метода суперэлементов. Программный комплекс КАСКАД-2. Сб.: Применение численных методов в строительной механике. – Л.: Судостроение, 1976.
49. Соловов А. В., Столярчук В. А. Автоматизированное проектирование силовых конструкций, МАИ, 1993.
50. Соловов А.В., Черепашков А.А. Подсистема проектирования силовых схем плоских конструкций учебной САПР ПРОСК, Куйбышев, КуАИ, 1986.

ДОДАТОК А

Лістинг коду

```
unit Main;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ComCtrls, ToolWin, ExtCtrls, StdCtrls, ImgList, Registry, IdGlobal,
  OleServer, IniFiles, WordXP, Variants, Clipbrd, Buffer, StrUtils;
type
  TMaterial = record
    MName: string;
    MModUp: extended;
    MDopNap: extended;
    MKoffPuas: extended;
    MPlotn: extended;
  end;
  TMain_Form = class(TForm)
    StatusBar1: TStatusBar;
    OpenDialog: TOpenDialog;
    MainToolBar_IL: TImageList;
    FermToolBar_IL: TImageList;
    New_PMnu: TPopupMenu;
    New_Btn_PU_Ferma: TMenuItem;
    New_Btn_PU_Plac: TMenuItem;
    PlacToolBar_IL: TImageList;
    Ferma_Graph_Enter_IL: TImageList;
    Main_Panel: TPanel;
    Main_ToolBar: TToolBar;
    New_Ferma_TBtn: TToolButton;
    Open_TBtn: TToolButton;
    Save_TBtn: TToolButton;
    TOKToolBar_IL: TImageList;
    Plac_Graph_Enter_IL: TImageList;
    TOK_Graph_Enter_IL: TImageList;
    TOK_PM: TPopupMenu;
    TOK_NO_PMI: TMenuItem;
    TOK_OK_PMI: TMenuItem;
    TOK_OpenDialog: TOpenDialog;
    SimpleSolve_List: TPopupMenu;
    Opt_solve_list: TPopupMenu;
    Plac_SimpReztext: TMenuItem;
    N8: TMenuItem;
    SimpleSolve_eq: TMenuItem;
    SimpleSolve_X: TMenuItem;
    SimpleSolve_Y: TMenuItem;
    SimpleSolve_Kas: TMenuItem;
    Plac_OptRezText: TMenuItem;
    N7: TMenuItem;
    Plac_TolRezGraph: TMenuItem;
    New_Btn_PU_TOK: TMenuItem;
    Ferma_Graph_Enter_Panel: TPanel;
```

Ferma_Graph_Enter_ToolBar: TToolBar;
None_ToolButton: TToolButton;
ToolButton5: TToolButton;
DrawPivot_ToolButton: TToolButton;
DeletePivot_ToolButton: TToolButton;
DeletePivot12_ToolButton: TToolButton;
ToolButton4: TToolButton;
Node_ToolButton: TToolButton;
NodeDelete_ToolButton: TToolButton;
ToolButton8: TToolButton;
Coord_ToolButton: TToolButton;
PivotTol_ToolButton: TToolButton;
ToolButton10: TToolButton;
ToolButton12: TToolButton;
Size_ToolButton: TToolButton;
Plast_Graph_Enter_ToolBar: TToolBar;
ToolButton1: TToolButton;
ToolButton2: TToolButton;
Cut_Plast_Toolbutton: TToolButton;
Del_Cut_Plast_Toolbutton: TToolButton;
ToolButton28: TToolButton;
Plast_Zakr_ToolButton: TToolButton;
ToolButton30: TToolButton;
Inform_Toolbutton: TToolButton;
ToolButton32: TToolButton;
plast_Size_ToolButton: TToolButton;
ToolButton3: TToolButton;
ToolButton6: TToolButton;
ToolButton27: TToolButton;
Tok_Inform_Toolbutton: TToolButton;
ToolButton31: TToolButton;
Ferma_Panel: TPanel;
Plast_Panel: TPanel;
Ferm_ToolBar: TToolBar;
ToolButton9: TToolButton;
FermaKonButton: TToolButton;
FermaGraphButton: TToolButton;
FermaNumberButton: TToolButton;
ToolButton15: TToolButton;
ToolButton16: TToolButton;
ToolButton17: TToolButton;
ToolButton18: TToolButton;
ToolButton19: TToolButton;
SimpleResult_TB: TToolButton;
ToolButton21: TToolButton;
ToolButton14: TToolButton;
ToolButton7: TToolButton;
Plast_ToolBar: TToolBar;
ToolButton13: TToolButton;
PlastKonButton: TToolButton;
PlastGraphButton: TToolButton;
PlastNumberButton: TToolButton;

ToolButton20: TToolButton;
 Plast_SN_Cbx: TComboBox;
 ToolButton23: TToolButton;
 SimpleSolve: TToolButton;
 ToolButton25: TToolButton;
 ToolButton26: TToolButton;
 SimpleRez_text: TToolButton;
 OptRez_text: TToolButton;
 ToolButton29: TToolButton;
 SimpleRez_graf: TToolButton;
 ToolButton11: TToolButton;
 TokKonButton: TToolButton;
 TokGraphButton: TToolButton;
 TokNumberButton: TToolButton;
 ToolButton22: TToolButton;
 tok_sn_cbx: TComboBox;
 ToolButton24: TToolButton;
 toksolve_btn: TToolButton;
 ToolButton34: TToolButton;
 tokrez_btn: TToolButton;
 F_new_Ferma_tbtn: TToolButton;
 F_Save_TBtn: TToolButton;
 ToolButton36: TToolButton;
 P_Save_TBtn: TToolButton;
 p_new_Ferma_tbtn: TToolButton;
 ToolButton39: TToolButton;
 ToolButton40: TToolButton;
 t_new_tok_tbtn: TToolButton;
 T_Save_TBtn: TToolButton;
 F_new_pmnu: TPopupMenu;
 f_New_Btn_PU_Ferma: TMenuItem;
 f_New_Btn_PU_TOK: TMenuItem;
 p_new_pmnu: TPopupMenu;
 p_New_Btn_PU_Ferma: TMenuItem;
 p_New_Btn_PU_TOK: TMenuItem;
 t_new_pmnu: TPopupMenu;
 t_New_Btn_PU_Ferma: TMenuItem;
 t_New_Btn_PU_TOK: TMenuItem;
 t_New_Btn_PU_Plac: TMenuItem;
 Nagr_ToolButton: TToolButton;
 Zak_ToolButton: TToolButton;
 Timer1: TTimer;
 PopupMenu1: TPopupMenu;
 N3: TMenuItem;
 N4: TMenuItem;
 Sn_CBx: TComboBox;
 ToolButton33: TToolButton;
 LoadOtr: TToolButton;
 OtrOpenDialog: TOpenDialog;
 OpenDialog1: TOpenDialog;
 PopupMenu2: TPopupMenu;
 N15: TMenuItem;

```

N16: TMenuItem;
PopupMenu3: TPopupMenu;
N17: TMenuItem;
MainMenu1: TMainMenu;
File1: TMenuItem;
New1: TMenuItem;
NewFerma_Mnu: TMenuItem;
NewTOK_Mnu: TMenuItem;
Open1: TMenuItem;
N2: TMenuItem;
Exit1: TMenuItem;
Windows_Mnu: TMenuItem;
N5: TMenuItem;
N13: TMenuItem;
N14: TMenuItem;
ToolButton37: TToolButton;
ToolButton38: TToolButton;
ToolButton41: TToolButton;
ToolButton43: TToolButton;
N19: TMenuItem;
Node_Move_ToolButton: TToolButton;
FermPrevBtn: TToolButton;
FermNextBtn: TToolButton;
TokPrevBtn: TToolButton;
TokNextBtn: TToolButton;
N20: TMenuItem;
ToolButton44: TToolButton;
ToolButton45: TToolButton;
N18: TMenuItem;
N21: TMenuItem;
    procedure Exit1Click(Sender: TObject);
    procedure Open1Click(Sender: TObject);
    procedure NewFerma_MnuClick(Sender: TObject);
    procedure Sn_CBxChange(Sender: TObject);
    procedure Open_TBtnClick(Sender: TObject);
    procedure Save_TBtnClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FermaGraphButtonClick(Sender: TObject);
    procedure DrawPivot_ToolButtonClick(Sender: TObject);
    procedure None_ToolButtonClick(Sender: TObject);
    procedure Coord_ToolButtonClick(Sender: TObject);
    procedure ToolButton10Click(Sender: TObject);
    procedure FermaKonButtonClick(Sender: TObject);
    procedure Node_ToolButtonClick(Sender: TObject);
    procedure PlastGraphButtonClick(Sender: TObject);
    procedure FermaNumberButtonClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
    procedure NodeDelete_ToolButtonClick(Sender: TObject);
    procedure Size_ToolButtonClick(Sender: TObject);
    procedure DeletePivot_ToolButtonClick(Sender: TObject);
    procedure PivotTol_ToolButtonClick(Sender: TObject);

```



```

procedure DeletePivot12_ToolButtonClick(Sender: TObject);
procedure N5Click(Sender: TObject);
procedure ToolButton17Click(Sender: TObject);
procedure ToolButton18Click(Sender: TObject);
procedure SimpleResult_TBClick(Sender: TObject);
procedure ToolButton21Click(Sender: TObject);
procedure ToolButton7Click(Sender: TObject);
procedure FormResize(Sender: TObject);
procedure ContentsClick(Sender: TObject);
procedure Del_Cut_Plast_ToolbuttonClick(Sender: TObject);
procedure SimpleSolveClick(Sender: TObject);
procedure ToolButton25Click(Sender: TObject);
procedure SimpleRez_textClick(Sender: TObject);
procedure SimpleRez_grafClick(Sender: TObject);
procedure OptRez_textClick(Sender: TObject);
procedure OptRez_grafClick(Sender: TObject);
procedure Tok_Size_ToolButtonClick(Sender: TObject);
procedure tok_Zakr_ToolButtonClick(Sender: TObject);
procedure ToolButton3Click(Sender: TObject);
procedure ToolButton1Click(Sender: TObject);
procedure toksolve_btnClick(Sender: TObject);
procedure tokrez_btnClick(Sender: TObject);
procedure Tok_Inform_ToolbuttonClick(Sender: TObject);
procedure Nagr_ToolButtonClick(Sender: TObject);
procedure Zak_ToolButtonClick(Sender: TObject);
procedure FormActivate(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure newClick(Sender: TObject);
procedure ToolButton33Click(Sender: TObject);
procedure N3Click(Sender: TObject);
procedure N4Click(Sender: TObject);
procedure LoadOtrClick(Sender: TObject);
procedure N15Click(Sender: TObject);
procedure N17Click(Sender: TObject);
procedure ToolButton37Click(Sender: TObject);
procedure ToolButton42Click(Sender: TObject);
procedure ToolButton38Click(Sender: TObject);
procedure ToolButton43Click(Sender: TObject);
procedure ToolButton41Click(Sender: TObject);
procedure MouseLeft(Form: TForm);
procedure MouseIsDown(Form: TForm; MX,MY: Integer);
procedure MouseIsMove(Form: TForm; MX,MY: Integer);
procedure MouseIsUp(Form: TForm; MX,MY: Integer);
procedure SelectionMode(Form: TForm; Activate: Boolean);
procedure LoadProfileClick(Sender: TObject);
procedure ModuleExecuteClick(Sender: TObject);
procedure N14Click(Sender: TObject);
procedure N13Click(Sender: TObject);
procedure N19Click(Sender: TObject);
procedure Node_Move_ToolButtonClick(Sender: TObject);
procedure FermPrevBtnClick(Sender: TObject);
procedure FermNextBtnClick(Sender: TObject);

```

```

procedure N20Click(Sender: TObject);
procedure Sort(Sender: TObject);
protected
  procedure WMDropFiles(var Message:TMessage); message WM_DROPFILES;
private
  FOnDropFiles:TNotifyEvent;
  FDrop:THandle;
  procedure DoDropFiles(Sender:TObject);
  { Private declarations }
public
  { Public declarations }
property Drop:THandle read FDrop write FDrop;
property OnDropFiles:TNotifyEvent read FOnDropFiles write FOnDropFiles;
  private
  public
    { Public declarations }
    User_Height: integer; // Высота экрана пользователя в пикселях
    User_Width: integer; // Ширина экрана пользователя в пикселях
    Height_Ratio: real; // Коэффициент высоты экрана. Определяется
                        // как отношение высоты экрана разработчика
                        // к высоте экрана пользователя
    Width_Ratio: real; // Коэффициент ширины экрана. Определяется
                      // как отношение ширины экрана разработчика
                      // к ширине экрана пользователя
    TileWindows: boolean;
    CascadeWindows:boolean; // Упорядочиваем ли мы окна ?
    Exit_Ok:boolean; // Выходим ли мы из программы ? (окончательно)
    Main_Window_Exit:boolean; //Из чего мы запустили процедуру Close
    output_order:integer; // переменная, определяющая порядок расстановки окон
    sorted:boolean;
    {Список Материалов для фермы}
    Ferma_MMaterials:array[1..100] of TMaterial; // Массив материалов для фермы
    Ferma_num_mat:integer; //Число добавленных пользователем материалов
    Num_Nonamed_Ferma:integer; //Номер вновь создаваемого проекта 'Ферма'
end;
var
  Main_Form: TMain_Form;
  keyid: integer;
  MouseX1,MouseY1,MouseX2,MouseY2: Integer;
  MDown: Boolean;
  Selection: Boolean;
  SelectionIsOK: Boolean;
  SelectionIsCopied: Boolean;
  ControlsArray: array of boolean;
  FormBMP: TBitmap;
  BStyle: TFormBorderStyle;
  BIcons: TBorderIcons;
  FSize: array [0..1] of Integer;
  ASize: Boolean;
  const
    crDeleteElement = 1; // Курсорчик удаления
    Dev_Height : Integer = 1024; // Высота экрана разработчика в пикселях

```

Dev_Width : Integer = 1280; // Ширина экрана разработчика в пикселях

implementation

```
uses Ferma_M, Plast_M, TOK_M, Ferm_Dat,
    Ferma_FD, FermaRegionSize, SimplRezFerm, DeFormFerma, Plast_FD, Visio, SelectMetod,
    ModuleExecute, ShellAPI;
{$R *.DFM}
{$R FermaPlus.res}
procedure TMain_Form.WMDropFiles(var Message:TMessage);
begin
    Drop:=Message.WParam;
    if Assigned(OnDropFiles) then OnDropFiles(Self);
end;
procedure TMain_Form.DoDropFiles(Sender:TObject);
var CB:Integer;filesCount,j:Integer;
    filePath:Array[0..MAX_PATH] of Char;
    str: String;
begin
    filesCount:=DragQueryFile((Sender as TMain_Form).Drop,$FFFFFFFF,nil,cb);
    for j:=0 to filesCount-1 do begin
        DragQueryFile((Sender as TMain_Form).Drop,j,filePath,MAX_PATH);
        str:=ExtractFileExt(LowerCase(filePath));
        if str='.frm' then
            begin
                with TFerma_Form.OpenFile(Self,filePath) do
                begin
                    Show;
                end;
            end
        else if str='.dnp' then
            begin
                with TPlast_Form.OpenFile(Self,filePath) do
                begin
                    Show;
                end;
            end
        else if str='.tok' then
            begin
                with Ttok_Form.OpenFile(Self,filePath) do
                begin
                    Show;
                end;
            end
        else begin
            MessageDlg('Неизвестный тип файла.',mtError,[mbOk],0);
        end;
    end;
end;
procedure TMain_Form.Exit1Click(Sender: TObject);
begin
    Close;
end;
```

```

procedure TMain_Form.Open1Click(Sender: TObject);
var
  str: String;
  already_open: boolean;
  i,j: integer;
  OpenFiles: array[0..11] of string;
begin
  if OpenFileDialog.Execute then
  for i:=0 to OpenFileDialog.Files.count-1 do begin
    already_open := False;
    for j := MDIChildCount-1 downto 0 do
    begin
      if Main_Form.MDIChildren[j].Caption = ExtractFileName(OpenDialog.Files[i]) then
        already_open := True;
    end;
    if already_open then continue;
    str:=ExtractFileExt(LowerCase(OpenDialog.Files[i]));
    if str='.frm' then
    begin
      OpenFiles[i]:=ExtractFileName(ChangeFileExt(OpenDialog.Files[i], ''))+ '_tmp'+'.frm';
      Windows.CopyFile(PChar(OpenDialog.Files[i]), PChar(OpenFiles[i]), true);
      with TFerma_Form.OpenFile(Self,OpenFiles[i]) do
      begin
        Show;
      end;
      F_Save_TBtn.Enabled:=False;
    end
    else if str='.dnp' then
    begin
      with TPlast_Form.OpenFile(Self,OpenDialog.Files[i]) do
      begin
        Show;
      end;
      P_Save_TBtn.Enabled:=False;
    end
    else if str='.tok' then
    begin
      with Ttok_Form.OpenFile(Self,OpenDialog.Files[i]) do
      begin
        Show;
      end;
      T_Save_TBtn.Enabled:=False;
    end
    else begin
      MessageDlg('Неизвестный тип файла.',mtError,[mbOk],0);
    end;
  end;
end;
procedure TMain_Form.NewFerma_MnuClick(Sender: TObject);
var
  F:TFerm;
begin

```

```

    F:=TFerm.Create;
    TFerma_Form.NewFile(Self,F);
end;
procedure TMain_Form.Sn_CBxChange(Sender: TObject);
begin
    ActiveMDIChild.Tag:=Sn_CBx.ItemIndex+1;
    ferma_FD_form.showD(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).Ferm);
    ActiveMDIChild.RePaint;
    if ((Main_Form.DeletePivot12_ToolButton.Down) or
(Main_Form.DeletePivot_ToolButton.Down)or(Main_Form.PivotTol_ToolButton.Down)) then
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdent:=0;
end;
procedure TMain_Form.Open_TBtnclick(Sender: TObject);
begin
    Open1Click(Self);
end;
procedure TMain_Form.Save_TBtnclick(Sender: TObject);
begin
    if ActiveMDIChild is TFerma_Form then
    begin
        (ActiveMDIChild as TFerma_Form).FileSave_MnuClick(Sender);
        F_Save_TBtnc.Enabled:=False;
    end;
    if ActiveMDIChild is TPlast_Form then
    begin
        (ActiveMDIChild as TPlast_Form).FileSave_MnuClick(Sender);
        P_Save_TBtnc.Enabled:=False;
    end;
    if ActiveMDIChild is Ttok_Form then
    begin
        (ActiveMDIChild as Ttok_Form).FileSave_MnuClick(Sender);
        T_Save_TBtnc.Enabled:=False;
    end;;
end;
procedure TMain_Form.NewPlast_MnuClick(Sender: TObject);
var
    p:TPlast;
begin
    p:=TPlast.Create;
    TPlast_Form.NewFile(Self,p);
end;
procedure TMain_Form.FormCreate(Sender: TObject);
begin
    with Screen do
    begin
        Height_Ratio    := Dev_Height/Height;
        Width_Ratio:= Dev_Width/Width;
        Main_Form.Width:= trunc(640*Width_Ratio);
        Main_Form.Height:= trunc(480*Height_Ratio);
    end;
    Application.HelpFile:=ExtractFilePath(Application.ExeName)+Application.HelpFile;

```

```

Main_Panel.Visible :=TRUE;
Num_Nonamed_Ferma :=1;
Num_Nonamed_Plast :=1;
Num_Nonamed_TOK :=1;
output_order := 1; //прямой порядок вывода
sorted:=false;
Main_Form.OnDropFiles:=Self.DoDropFiles;
DragAcceptFiles(Main_Form.Handle,True);
end;
procedure TMain_Form.FermaKonButtonClick(Sender: TObject);
begin
    if FermaKonButton.Down = False then ferma_FD_form.Visible:=False
    else ferma_FD_form.Visible:=TRUE;
end;
procedure TMain_Form.FermaGraphButtonClick(Sender: TObject);
begin
    if FermaGraphButton.Down = False then
    begin
        Ferma_Graph_Enter_Panel.Visible :=False;
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).ViewGraph_Mnu.Checked := False;
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crDefault;
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint :=False;
        Ferma_Graph_Enter_ToolBar.Buttons[0].Down:=TRUE;
        StatusBar1.Panels[0].Text:= "";
        StatusBar1.Panels[1].Text:= "";
        StatusBar1.Panels[2].Text:= "";
    end
    else begin
        Ferma_Graph_Enter_Panel.Visible :=TRUE;
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).ViewGraph_Mnu.Checked :=
TRUE;
    end;
end;
procedure TMain_Form.DrawPivot_ToolButtonClick(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crCross;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
    Main_Form.StatusBar1.Panels[0].Text := '[- : -]';
    Main_Form.StatusBar1.Panels[1].Text := 'Вне пределов области';
    Main_Form.StatusBar1.Panels[2].Text := 'Переместите курсор в выделенную область';
    if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdent<>0) then
    begin
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdent:=0;
        Main_Form.Node_Move_ToolButton.Down:=false;
    end;
end;
procedure TMain_Form.None_ToolButtonClick(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crDefault;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
    Main_Form.StatusBar1.Panels[0].Text := "";

```

```

Main_Form.StatusBar1.Panels[1].Text := "";
Main_Form.StatusBar1.Panels[2].Text := "";
if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet<>0) then
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet:=0;
    Main_Form.Node_Move_ToolButton.Down:=false;
end;
end;
procedure TMain_Form.Coord_ToolButtonClick(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crHandPoint;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
    Main_Form.StatusBar1.Panels[0].Text := "";
    Main_Form.StatusBar1.Panels[1].Text := "";
    Main_Form.StatusBar1.Panels[2].Text := "";
    if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet<>0) then
    begin
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet:=0;
    end;
    Main_Form.Node_Move_ToolButton.Down:=false;
end;
procedure TMain_Form.Zak_ToolButtonClick(Sender: TObject);
begin
    Main_Form.Node_Move_ToolButton.Down:=false;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crHandPoint;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
    Main_Form.StatusBar1.Panels[0].Text := "";
    Main_Form.StatusBar1.Panels[1].Text := "";
    Main_Form.StatusBar1.Panels[2].Text := "";
    if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet<>0) then
    begin
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet:=0;
    end;
end;
procedure TMain_Form.Nagr_ToolButtonClick(Sender: TObject);
begin
    Main_Form.Node_Move_ToolButton.Down:=false;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crHandPoint;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
    Main_Form.StatusBar1.Panels[0].Text := "";
    Main_Form.StatusBar1.Panels[1].Text := "";
    Main_Form.StatusBar1.Panels[2].Text := "";
    if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet<>0) then
    begin
        Ferma_M.TFerma_Form(Main_For.ActiveMDIChild).PivotIdet:=0;
    end;
end;
end;
procedure TMain_Form.ToolButton10Click(Sender: TObject);
begin

```

```

Main_Form.Node_Move_ToolButton.Down:=false;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crHelp;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=TRUE;
Main_Form.StatusBar1.Panels[0].Text := "";
Main_Form.StatusBar1.Panels[1].Text := "";
Main_Form.StatusBar1.Panels[2].Text := "";
if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdent<>0) then
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdent:=0;
end;
end;
procedure TMain_Form.Node_ToolButtonClick(Sender: TObject);
begin
    Main_Form.Node_Move_ToolButton.Down:=false;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crCross;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
    Main_Form.StatusBar1.Panels[0].Text := '[- : -]';
    Main_Form.StatusBar1.Panels[1].Text := 'Вне пределов области';
    Main_Form.StatusBar1.Panels[2].Text := 'Переместите курсор в выделенную область';
    if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdent<>0) then
    begin
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdent:=0;
    end;
end;

procedure TMain_Form.PlastGraphButtonClick(Sender: TObject);
begin

    if PlastGraphButton.Down = False then
    begin
        Plast_Graph_Enter_Panel.Visible          :=False;
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).Plast_ViewGraph_Mnu.Checked :=
False;
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).paintbox.Cursor          :=crDefault;
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint        :=False;
        Plast_Graph_Enter_ToolBar.Buttons[0].Down    :=TRUE;
        StatusBar1.Panels[0].Text := "";
        StatusBar1.Panels[1].Text := "";
        StatusBar1.Panels[2].Text := "";
    end
    else begin
        Plast_Graph_Enter_Panel.Visible          :=TRUE;
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).Plast_ViewGraph_Mnu.Checked :=
TRUE;
    end;
end;
procedure TMain_Form.FermaNumberButtonClick(Sender: TObject);
var
    I:integer;
begin

```



```

    if FermaNumberButton.Down=False then
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).Num_Element.Checked:=False
    else Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).Num_Element.Checked:=TRUE;
    for I := MDIChildCount-1 downto 0 do
        Main_Form.MDIChildren[I].RePaint;

end;
procedure TMain_Form.FormClose(Sender: TObject; var Action: TCloseAction);
var
    I:integer;
    Ferma_File_Ok:boolean; //Флажок для проверки наличия файла с материалами
    // для фермы
    Ferma_mf:System.Text; // Файл с материалами для фермы
    Plast_File_Ok:boolean;
    Plast_mf:System.Text;
    TOK_File_Ok:boolean; //Флажок для проверки наличия файла с материалами
    // для TOK
    TOK_mf:System.Text; //
begin
    Exit_Ok      :=TRUE;
    Main_Window_Exit:=TRUE;
    Action       :=caFree;

    for I := MDIChildCount-1 downto 0 do
    begin
        Main_Form.MDIChildren[I].Close;
        if not Exit_Ok then break;
    end;

    if not Exit_Ok then
    begin
        Action       :=caNone;
        Main_Window_Exit := False;
        exit;
    end;

    ferma_FD_form.Close;
    Plast_Fd_Form.Close;
    TOK_Fd_Form.Close;
    //===== Обработка файла с материалами для фермы =====
    Ferma_File_Ok:=TRUE;
    AssignFile(Ferma_mf,ExtractFilePath(Application.ExeName)+'fermamaterials.ini');
    if      FileExists(ExtractFilePath(Application.ExeName)+'fermamaterials.ini')      then
reset(Ferma_mf)
    else Ferma_File_Ok:=False;
    if Ferma_num_mat=0 then // Количество материалов в файле для фермы
    begin
        if Ferma_File_Ok then
        begin CloseFile(Ferma_mf);//Erase(Ferma_mf);
            end; // Если файл есть, а все дополнительные материалы удалены, то файл стираем
        end
    else begin

```

```

        rewrite(Ferma_mf);
writeln(Ferma_mf,Ferma_num_mat);
for I:=3 to Ferma_num_mat+2 do
begin
    writeln(Ferma_mf,Ferma_MMaterials[I].MName);
    writeln(Ferma_mf,Ferma_MMaterials[I].MModUpr);
    writeln(Ferma_mf,Ferma_MMaterials[I].MDopNaprr);
    writeln(Ferma_mf,Ferma_MMaterials[I].Mplotn);
end;
CloseFile(Ferma_mf);
// end;
end;
//===== Конец обработки файла с материалами для фермы =====

Plast_File_Ok:=TRUE;

AssignFile(Plast_mf,ExtractFilePath(Application.ExeName)+'plastinamaterials.ini');
if FileExists(ExtractFilePath(Application.ExeName)+'plastinamaterials.ini') then
begin
    reset(Plast_mf);
end
else Plast_File_Ok:=False;
if Plast_num_mat=0 then //
begin
    if Plast_File_Ok then
    begin CloseFile(Plast_mf);Erase(Plast_mf);
    end; // Если файл есть, а все дополнительные материалы удалены, то файл стираем
end
else begin
    rewrite(Plast_mf);
    writeln(Plast_mf,Plast_num_mat);
    for I:=3 to Plast_num_mat+2 do
    begin
        writeln(Plast_mf,Plast_MMaterials[I].MName);
        writeln(Plast_mf,Plast_MMaterials[I].MModUpr);
        writeln(Plast_mf,Plast_MMaterials[I].MDopNaprr);
        writeln(Plast_mf,Plast_MMaterials[I].MKoffPuas);
        writeln(Plast_mf,Plast_MMaterials[I].MPlotn);
    end;
    CloseFile(Plast_mf);
    // end;
end;
end;
procedure TMain_Form.FormShow(Sender: TObject);
var
    Ferma_mf:System.Text; // Файл с материалами для фермы
    Ferma_File_Ok:boolean; //Флажок для проверки наличия файла с материалами
    // для фермы
    Plast_mf:System.Text;
    Plast_File_Ok:boolean;
    TOK_mf:System.Text;

```

```

TOK_File_Ok:boolean; //Флажок для проверки наличия файла с материалами
// для TOK
I:integer;
begin
//Устанавливаем необходимые курсоры из ресурса
Screen.Cursors[crDeleteElement] := LoadCursor(HInstance, 'DELETEELEMENT');
Ferma_File_Ok      :=TRUE;
Ferma_num_mat      :=0;
// Устанавливаем материалы встроенные в программу (их нельзя удалить)
Ferma_MMaterials[1].MName      :='Сталь';
Ferma_MMaterials[1].MModUpr    :=20000000;
Ferma_MMaterials[1].MDopNapra  :=70000;
Ferma_MMaterials[2].MName      :='Алюминиевый сплав';
Ferma_MMaterials[2].MModUpr    :=7000000;
Ferma_MMaterials[2].MDopNapra  :=30000;
Ferma_MMaterials[1].MPlotn     :=7.8*0.001;
Ferma_MMaterials[2].MPlotn     :=2.8*0.001;
if FileExists(ExtractFilePath(Application.ExeName)+'fermamaterials.ini') then
begin
    AssignFile(Ferma_mf,ExtractFilePath(Application.ExeName)+'fermamaterials.ini');
    reset(Ferma_mf);
end
else Ferma_File_Ok:=False;
if Ferma_File_Ok then // Если файл есть, то считываем дополнительные материалы
begin
    readln(Ferma_mf,Ferma_num_mat);
    for I:=3 to Ferma_num_mat+2 do
    begin
        readln(Ferma_mf,Ferma_MMaterials[I].MName,
            Ferma_MMaterials[I].MModUpr,
            Ferma_MMaterials[I].MDopNapra,Ferma_MMaterials[I].MPlotn);
    end;
    CloseFile(Ferma_mf);
end;
Plast_File_Ok :=TRUE;
Plast_num_mat      :=0;
    Plast_MMaterials[1].MName      :='Сталь';
Plast_MMaterials[1].MModUpr :=20000000;
Plast_MMaterials[1].MDopNapra :=70000;
Plast_MMaterials[1].MKoffPuas:=0.3;
Plast_MMaterials[2].MName      :='Алюминиевый сплав';
Plast_MMaterials[2].MModUpr :=7000000;
Plast_MMaterials[2].MDopNapra :=30000;
Plast_MMaterials[2].MKoffPuas:=0.34;
Plast_MMaterials[1].MPlotn     :=7.8*0.001;
Plast_MMaterials[2].MPlotn     :=2.8*0.001;
if FileExists(ExtractFilePath(Application.ExeName)+'plastinamaterials.ini') then
begin
    AssignFile(Plast_mf,ExtractFilePath(Application.ExeName)+'plastinamaterials.ini');
    reset(Plast_mf);
end
else Plast_File_Ok:=False;

```

```

if Plast_File_Ok then // Если файл есть, то считываем дополнительные материалы
begin
  readln(Plast_mf,Plast_num_mat);
  for I:=3 to Plast_num_mat+2 do
  begin
    readln(Plast_mf,Plast_MMaterials[I].MName,
           Plast_MMaterials[I].MModUp,
           Plast_MMaterials[I].MDopNap,
           Plast_MMaterials[I].MKoffPuas,Plast_MMaterials[I].Mplotn);
  end;
  CloseFile(Plast_mf);
end;

end;
procedure TMain_Form.NodeDelete_ToolButtonClick(Sender: TObject);
begin
  Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crDeleteElement;
  Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
  Main_Form.StatusBar1.Panels[0].Text := "";
  Main_Form.StatusBar1.Panels[1].Text := "";
  Main_Form.StatusBar1.Panels[2].Text := "";
  if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet<>0) then
  begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet:=0;
  end;
  Main_Form.Node_Move_ToolButton.Down:=false;
end;
procedure TMain_Form.Size_ToolButtonClick(Sender: TObject);
var
  I:integer;
  max_x_coord, max_y_coord:extended;
  F:TFerm;
begin
  Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
  Main_Form.StatusBar1.Panels[0].Text := "";
  Main_Form.StatusBar1.Panels[1].Text := "";
  Main_Form.StatusBar1.Panels[2].Text := "";
  if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet<>0) then
  begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet:=0;
  end;
  Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crDefault;
  Main_Form.Ferma_Graph_Enter_ToolBar.Buttons[0].Down:=TRUE;
  F := Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).Ferm;
  FermaRegionSizeForm.SizeLabel.Caption :='Размер области ['+F.S_lin+']';
  max_x_coord :=0;
  max_y_coord :=0;
  for I:=1 to F.nyz1 do
  begin
    if F.corn[I,1]>=max_x_coord then max_x_coord:=F.corn[I,1];

```

```

    if F.corn[I,2]>=max_y_coord then max_y_coord:=F.corn[I,2];
end;
FermaRegionSizeForm.MinX_L.Caption:=FormatFloat('0.##',max_x_coord)+' <=';
FermaRegionSizeForm.MinY_L.Caption:=FormatFloat('0.##',max_y_coord)+' <=';
if max_x_coord=0 then
FermaRegionSizeForm.MinX_L.Caption:=FormatFloat('0.##',max_x_coord)+' <';
if max_y_coord=0 then
FermaRegionSizeForm.MinY_L.Caption:=FormatFloat('0.##',max_y_coord)+' <';
FermaRegionSizeForm.XSize.Text:=FloatToStr(F.region_x);
FermaRegionSizeForm.YSize.Text:=FloatToStr(F.region_y);
FermaRegionSizeForm.ShowModal;
end;
procedure TMain_Form.DeletePivot_ToolButtonClick(Sender: TObject);
begin
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crDeleteElement;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotId<>0) then
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotX1 :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotX2 :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotY1 :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotY2 :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotId :=0;
Main_Form.StatusBar1.Panels[0].Text := "";
Main_Form.StatusBar1.Panels[1].Text := "";
Main_Form.StatusBar1.Panels[2].Text := "";
Main_Form.Node_Move_ToolButton.Down:=false;
end;
procedure TMain_Form.PivotTol_ToolButtonClick(Sender: TObject);
begin
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crHandPoint;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotId<>0) then
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotX1 :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotX2 :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotY1 :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotY2 :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotId :=0;
Main_Form.StatusBar1.Panels[0].Text := "";
Main_Form.StatusBar1.Panels[1].Text := "";
Main_Form.StatusBar1.Panels[2].Text := "";
Main_Form.Node_Move_ToolButton.Down:=false;
end;
procedure TMain_Form.DeletePivot12_ToolButtonClick(Sender: TObject);
begin
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crDeleteElement;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotId<>0) then
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotX1 :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotX2 :=-1;

```

```

Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotY1      :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).pivotY2      :=-1;
Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdent    :=0;
Main_Form.StatusBar1.Panels[0].Text := "";
Main_Form.StatusBar1.Panels[1].Text := "";
Main_Form.StatusBar1.Panels[2].Text := "";
Main_Form.Node_Move_ToolButton.Down:=false;
end;
procedure TMain_Form.N5Click(Sender: TObject);
begin
    if not sorted then
        Main_Form.Sort(Sender);
    CascadeWindows:=TRUE;
    Cascade;
    CascadeWindows:=False;
    n5.Checked:=true;
    n13.Checked:=false;
    n14.Checked:=false;
end;
procedure TMain_Form.ToolButton17Click(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).SimpleSolve_MnuClick(Sender);
end;
procedure TMain_Form.ToolButton18Click(Sender: TObject);
begin
    if Ferma_SelectMetod=nil then
        Ferma_SelectMetod:=TFerma_SelectMetod.Create(self);

        Ferma_SelectMetod.Show;
end;
procedure TMain_Form.SimpleResult_TBClick(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).SimpleResults_MnuClick(Sender);
end;
procedure TMain_Form.ToolButton21Click(Sender: TObject);
begin
    case Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).last_opt_type of
        1:
            Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).N10Click(Sender);
        2:
            Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).OptResults_MnuClick(Sender);
        3:
            Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).N11Click(Sender);
        4:
            Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).N13Click(Sender);
    end;
end;
procedure TMain_Form.ToolButton7Click(Sender: TObject);
var
    FileName,Fname:string;
    Version_VYV,tmp:string;
    fl:System.Text;

```

```

ff:File of Byte;
mg:integer;
begin
FileName:=Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).FileName;
// Проверка на существование файла расчета
if FileExists(ChangeFileExt(FileName,'.vyv')) then
begin
AssignFile(f1,ChangeFileExt(FileName,'.vyv'));
reset(f1);
readln(f1,Version_VYV);
CloseFile(f1);
end
else Version_VYV:='Error';

if Version_VYV='Error' then
begin
Beep;
MessageDlg(#13+'Файл с простым расчетом не найден.',mtError,[mbOk],0);
exit;
end;
// Проверка на совместимость версий файлов данных и простого расчета
if FileExists(FileName) then
begin
AssignFile(ff,FileName);
reset(ff);
mg:=filesize(ff);
CloseFile(ff);
end
else begin
Beep;
MessageDlg('Несовпадение версий файлов с данными и простого
расчета.'+#13+'Произведите простой расчет для данной ферменной
конструкции.',mtError,[mbOk],0);
exit;
end;
if not AnsiContainsStr(Version_VYV,'_tmp') then
begin
Fname:= ExtractFileName(FileName);
Delete(Fname,Pos('_tmp',Fname),4);
tmp:= (Fname+' '+IntToStr(mg)+'
'+DateTimeToStr(FileDateToDateTime(FileAge(FileName))))
end
else
tmp:=(ExtractFileName(FileName)+' '+IntToStr(mg)+'
'+DateTimeToStr(FileDateToDateTime(FileAge(FileName)))));
if ((Version_VYV)<> tmp) or
(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).isChanged=TRUE) then
begin
Beep;
MessageDlg('Несовпадение версий файлов с данными и простого
расчета.'+#13+'Произведите простой расчет для данной ферменной
конструкции.',mtError,[mbOk],0);

```

```

        exit;
    end;
    //
    if DeForm_Form.Num_Color=0 then DeForm_Form.Num_Color:=5;
    DeForm_Form.Show;
end;
procedure TMain_Form.FormResize(Sender: TObject);
begin
    if Width<640 then Width:=640;
    if Height<450 then Height:=450;
end;
procedure TMain_Form.PlastKonButtonClick(Sender: TObject);
begin
    if PlastKonButton.Down = False then Plast_Fd_Form.Visible:=False
    else Plast_Fd_Form.Visible:=TRUE;
end;
procedure TMain_Form.Inform_ToolbuttonClick(Sender: TObject);
begin
    Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crHelp;
    Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=TRUE;
    Main_Form.StatusBar1.Panels[0].Text := "";
    Main_Form.StatusBar1.Panels[1].Text := "";
    Main_Form.StatusBar1.Panels[2].Text := "";

    if((Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).cutIdent<>"")or(Plast_M.TPlast_Form(
    Main_Form.ActiveMDIChild).XOld<>-
    1)or(Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).YOld<>-1)) then
        begin
            Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).RePaint;
            Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).cutIdent:="";
            Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).XOld :=-1;
            Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).YOld :=-1;
            Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).Zagr :=TRUE;
        end;
    end;
    procedure TMain_Form.SimpleSolveClick(Sender: TObject);
    begin
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).SimpleSolve_MnuClick(Sender);
    end;
    procedure TMain_Form.ToolButton25Click(Sender: TObject);
    begin
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).SolveOpt_MnuClick(Sender);
    end;
    procedure TMain_Form.Plast_SimpReztextClick(Sender: TObject);
    begin
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).s_text_mnuClick(Sender);
    end;
    procedure TMain_Form.Plast_OptRezTextClick(Sender: TObject);
    begin
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).o_text_mnuClick(Sender);
    end;
    procedure TMain_Form.Plast_TolRezGraphClick(Sender: TObject);

```



```

begin
  Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).OptResults_MnuClick(Sender);
end;
procedure TMain_Form.TokKonButtonClick(Sender: TObject);
begin
  if TokKonButton.Down = False then TOK_Fd_Form.Visible:=False
  else TOK_Fd_Form.Visible:=TRUE;
end;
procedure TMain_Form.TokNumberButtonClick(Sender: TObject);
var
  I:integer;
begin
  if TokNumberButton.Down=False then
  TOK_M.Ttok_Form(Main_Form.ActiveMDIChild).TOK_Num_Element.Checked:=False
  else
  TOK_M.Ttok_Form(Main_Form.ActiveMDIChild).TOK_Num_Element.Checked:=TRUE;

  for I := MDIChildCount-1 downto 0 do
    Main_Form.MDIChildren[I].Repaint;
  end;
procedure TMain_Form.SimpleRez_textClick(Sender: TObject);
begin
  Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).s_text_mnuClick(Sender);
end;
procedure TMain_Form.SimpleRez_grafClick(Sender: TObject);
begin
  Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).SimpleResults_MnuClick(Sender);
end;
procedure TMain_Form.OptRez_textClick(Sender: TObject);
begin
  Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).o_text_mnuClick(Sender);
end;
procedure TMain_Form.OptRez_grafClick(Sender: TObject);
begin
  Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).OptResults_MnuClick(Sender);
end;
procedure TMain_Form.Tok_Size_ToolButtonClick(Sender: TObject);
var
  I:integer;
  max_x_coord, max_y_coord:extended;
  t:Ttok;
begin
  TOK_M.Ttok_Form(Main_Form.ActiveMDIChild).paintbox.ShowHin:=False;
  Main_Form.StatusBar1.Panels[0].Text      := "";
  Main_Form.StatusBar1.Panels[1].Text      := "";
  Main_Form.StatusBar1.Panels[2].Text := "";
  TOK_M.Ttok_Form(Main_Form.ActiveMDIChild).paintbox.Cursor:=crDefault;
  Main_Form.TOK_Graph_Enter_ToolBar.Buttons[0].Down:=TRUE;
  t := TOK_M.Ttok_Form(Main_Form.ActiveMDIChild).TOK;
  tokRegionSizeForm.SizeLabel.Caption := 'Размеры области ['+t.S_lin+']';
  max_x_coord :=0;
  max_y_coord :=0;

```

```

for I:= 1 to 36 do
begin
    if t.xm[I]>max_x_coord then max_x_coord:=t.xm[I];
    if t.ym[I]>max_y_coord then max_y_coord:=t.ym[I];
end;
tokRegionSizeForm.MinX_L.Caption:=FormatFloat('0.##',max_x_coord)+' <=';
tokRegionSizeForm.MinY_L.Caption:=FormatFloat('0.##',max_y_coord)+' <=';
if max_x_coord=0 then
tokRegionSizeForm.MinX_L.Caption:=FormatFloat('0.##',max_x_coord)+' <';
if max_y_coord=0 then
tokRegionSizeForm.MinY_L.Caption:=FormatFloat('0.##',max_y_coord)+' <';
tokRegionSizeForm.XSize.Text:=FloatToStr(t.xm[37]);
tokRegionSizeForm.YSize.Text:=FloatToStr(t.ym[37]);
tokRegionSizeForm.ShowModal
end;
procedure TMain_Form.ToolButton3Click(Sender: TObject);
begin
    TOK_M.Ttok_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crDefault;
    TOK_M.Ttok_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint :=False;
    Main_Form.StatusBar1.Panels[0].Text := "";
    Main_Form.StatusBar1.Panels[1].Text := "";
    Main_Form.StatusBar1.Panels[2].Text := "";
end;
procedure TMain_Form.ToolButton1Click(Sender: TObject);
begin
    Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crDefault;
    Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
    Main_Form.StatusBar1.Panels[0].Text := "";
    Main_Form.StatusBar1.Panels[1].Text := "";
    Main_Form.StatusBar1.Panels[2].Text := "";

    if((Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).cutIdent<>")or(Plast_M.TPlast_Form(
Main_Form.ActiveMDIChild).XOld<>-
1)or(Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).YOld<>-1)) then
    begin
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).RePaint;
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).cutIdent:="";
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).XOld :=-1;
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).YOld :=-1;
        Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).Zagr :=TRUE;
    end;
end;
procedure TMain_Form.Tok_Inform_ToolbuttonClick(Sender: TObject);
begin
    TOK_M.Ttok_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crHelp;
    TOK_M.Ttok_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint :=TRUE;
    Main_Form.StatusBar1.Panels[0].Text := "";
    Main_Form.StatusBar1.Panels[1].Text := "";
    Main_Form.StatusBar1.Panels[2].Text := "";
end;
procedure TMain_Form.FormActivate(Sender: TObject);
begin

```

```

if MDIChildCount = 0 then Main_ToolBar.Buttons[4].enabled := False;
//showmessage(string(ParamStr(1)));
if Main_Form.Active then TFerma_Form.OpenFile(Main_Form,ParamStr(1));
if pos(string(ParamStr(1)), '.frm') <> 0 then
begin
    showmessage('fuck');
    TFerma_Form.OpenFile(Self,ParamStr(1));
end;
if Timer1<>nil then Timer1.enabled := TRUE;
end;

procedure TMain_Form.Timer1Timer(Sender: TObject);
begin
    if pos('frm',LowerCase(ParamStr(1))) <> 0 then
    TFerma_Form.OpenFile(Self,LowerCase(ParamStr(1)));
    if pos('dnp',LowerCase(ParamStr(1))) <> 0 then
    TPlast_Form.OpenFile(Self,LowerCase(ParamStr(1)));
    if pos('tok',LowerCase(ParamStr(1))) <> 0 then
    Ttok_Form.OpenFile(Self,LowerCase(ParamStr(1)));
    Timer1.Destroy;
end;
procedure TMain_Form.newClick(Sender: TObject);
begin
    if Ferma_SelectMetod=nil then
        Ferma_SelectMetod:=TFerma_SelectMetod.Create(self);
        Ferma_SelectMetod.Show;
end;

procedure TMain_Form.ToolButton33Click(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).OptResultsVC1_MnuClick(Sender);
end;
procedure TMain_Form.N3Click(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).OptResults_MnuClick(Sender);
end;

procedure TMain_Form.N4Click(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).OptResultsVC1_MnuClick(Sender);
end;
procedure TMain_Form.LoadOtrClick(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).LoadOtrData;
end;
procedure TMain_Form.N17Click(Sender: TObject);
begin
    if N17.Checked then
        N17.Checked:=False
    else
        N17.Checked:=True;
end;

```

```

procedure TMain_Form.ToolButton37Click(Sender: TObject);
begin
  Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PrepareForModule;
end;
procedure TMain_Form.ToolButton42Click(Sender: TObject);
begin
  Plast_M.TPlast_Form(Main_Form.ActiveMDIChild).PutToAccount;
end;
procedure TMain_Form.ToolButton43Click(Sender: TObject);
begin
  Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PrepareForModule;
end;
procedure TMain_Form.ToolButton41Click(Sender: TObject);
begin
  Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PrepareForModule;
end;
procedure TMain_Form.MouseLeft(Form: TForm);
begin
  Form.Repaint;
  Form.Canvas.CopyRect(Rect(0,0,FormBMP.Width,FormBMP.Height),
    FormBMP.Canvas,Rect(0,0,FormBMP.Width,FormBMP.Height));
end;
procedure TMain_Form.MouseIsDown(Form: TForm; MX,MY: Integer);
begin
  MDown:=True;
  MouseX1:=MX;
  MouseY1:=MY;
  Form.Canvas.Pen.Color:=clBlue;
  MouseLeft(Form);
end;
procedure TMain_Form.MouseIsMove(Form: TForm; MX,MY: Integer);
begin
  if MDown then
  begin
    Form.Repaint;
    with Form.Canvas do
    begin
      CopyRect(Rect(0,0,FormBMP.Width,FormBMP.Height),
        FormBMP.Canvas,Rect(0,0,FormBMP.Width,FormBMP.Height));
      MoveTo(MouseX1,MouseY1);
      LineTo(MX,MouseY1);
      LineTo(MX,MY);
      LineTo(MouseX1,MY);
      LineTo(MouseX1,MouseY1);
    end;
  end;
end;
procedure TMain_Form.MouseIsUp(Form: TForm; MX,MY: Integer);
var
  Temp: Integer;
begin
  MDown:=False;

```

```

MouseX2:=MX;
MouseY2:=MY;
if MouseX2<MouseX1 then
begin
    Temp:=MouseX1;
    MouseX1:=MouseX2;
    MouseX2:=Temp;
end;
if MouseY2<MouseY1 then
begin
    Temp:=MouseY1;
    MouseY1:=MouseY2;
    MouseY2:=Temp;
end;
if (MouseX1<>MouseX2) or (MouseY1<>MouseY2) then SelectionIsOK:=True
else SelectionIsOK:=False;
end;
procedure TMain_Form.SelectionMode(Form: TForm; Activate: Boolean);
var
    BMP: TBitmap;
    i,j: Integer;
begin
    // Если включаем режим:
    if Activate then
    begin
        if Form.AutoSize then
        begin
            Form.AutoSize:=False;
            ASize:=True;
        end
        else
            ASize:=False;
        BMP:=TBitmap.Create;
        BMP.Width:=Form.ClientWidth;
        BMP.Height:=Form.ClientHeight;
        FormBMP.Width:=BMP.Width;
        FormBMP.Height:=BMP.Height;
        BMP.Canvas.CopyRect(Rect(0,0,BMP.Width,BMP.Height),Form.Canvas,
            Rect(0,0,BMP.Width,BMP.Height));
        SetLength(ControlsArray,Form.ControlCount);
        for i:=0 to Form.ControlCount-1 do
        begin
            if Form.Controls[i].Visible then
            begin
                Form.Controls[i].Visible:=False;
                ControlsArray[i]:=True;
            end
            else ControlsArray[i]:=False;
        end;
        FSize[0]:=Form.Width;
        FSize[1]:=Form.Height;
        BStyle:=Form.BorderStyle;
    end;
end;

```

```

Form.BorderStyle:=bsSingle;
BIcons:=Form.BorderIcons;
Form.BorderIcons:=[biSystemMenu];
Form.Width:=FSize[0];
Form.Height:=FSize[1];
Form.Canvas.CopyRect(Rect(0,0,BMP.Width,BMP.Height),BMP.Canvas,
  Rect(0,0,BMP.Width,BMP.Height));
FormBMP.Canvas.CopyRect(Rect(0,0,BMP.Width,BMP.Height),BMP.Canvas,
  Rect(0,0,BMP.Width,BMP.Height));
BMP.Free;
end
// Или же если выключаем:
else
begin
  for i:=0 to Form.ControlCount-1 do
  begin
    if ControlsArray[i]=True then
    begin
      Form.Controls[i].Visible:=True;
    end;
  end;
  Form.Repaint;
  FormBMP.Width:=0;
  FormBMP.Height:=0;
  SetLength(ControlsArray,0);
  FSize[0]:=Form.Width;
  FSize[1]:=Form.Height;
  Form.BorderStyle:=BStyle;
  Form.BorderIcons:=BIcons;
  Form.Width:=FSize[0];
  Form.Height:=FSize[1];
  if ASize then
  begin
    Form.AutoSize:=True;
    ASize:=False;
  end;
end;
end;
procedure TMain_Form.ModuleExecuteClick(Sender: TObject);
begin
  ModuleExecute_Form.ShowModal;
end;
procedure TMain_Form.N14Click(Sender: TObject);
begin
  if not sorted then
    Main_Form.Sort(Sender);
  TileWindows:=True;
  TileMode:=tbVertical;
  Tile;
  TileWindows:=False;
  n14.Checked:=true;
  n5.Checked:=false;

```

```

n13.Checked:=false;
end;
procedure TMain_Form.N13Click(Sender: TObject);
begin
  if not sorted then
    Main_Form.Sort(Sender);
  TileWindows:=True;
  TileMode:=tbHorizontal;
  Tile;
  TileWindows:=False;
  n13.Checked:=true;
  n5.Checked:=false;
  n14.Checked:=false;
end;
procedure TMain_Form.N19Click(Sender: TObject);
var
  i,j: integer;
  s,ss: string;
  t: TStringList;
  found: integer;
begin
  t := TStringList.Create;
  for i:=1 to Main_form.MDICHildCount do begin
    t.Add(Main_form.MDICHildren[i-1].Caption);
    //Main_form.MDICHildren[i-1].SetFocus;
  end;
  t.Sort;
  s:="";
  ss:='list: ';
  if output_order=1 then
    begin
      for j:=0 to t.Count-1 do begin
        ss:= ss+ ' '+t.Strings[j];
      end;
      for j:=0 to t.Count-1 do begin
        for i:=1 to Main_form.MDICHildCount do begin
          found:=0;
          if (found=0) then begin
            if (Main_form.MDICHildren[i-1].Caption=t.Strings[j]) then begin
              Main_form.MDICHildren[i-1].SetFocus;
              found:=1;
            end;
          end;
        end;
      end;
      output_order:=2;
    end
  else begin
    for j:=t.Count-1 downto 0 do begin
      for i:=1 to Main_form.MDICHildCount do begin
        found:=0;
        if (found=0) then begin

```

```

        if (Main_form.MDICHildren[i-1].Caption=t.Strings[j]) then begin
            Main_form.MDICHildren[i-1].SetFocus;
            found:=1;
        end;
    end;
end;
end;
output_order:=1;
end;
if n5.Checked then n5.Click;
if n13.Checked then n13.Click;
if n14.Checked then n14.Click;
if not n5.Checked then begin
    if not n13.Checked then begin
        if not n14.Checked then begin
            n5.Click;
        end;
    end;
end;
end;
procedure TMain_Form.Node_Move_ToolButtonClick(Sender: TObject);
begin
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.Cursor :=crDefault;
    Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).paintbox.ShowHint:=False;
    Main_Form.StatusBar1.Panels[0].Text := "";
    Main_Form.StatusBar1.Panels[1].Text := "";
    Main_Form.StatusBar1.Panels[2].Text := "";
    if(Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet<>0) then
    begin
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).RePaint;
        Ferma_M.TFerma_Form(Main_Form.ActiveMDIChild).PivotIdet:=0;
    end;
end;
procedure TMain_Form.FermPrevBtnClick(Sender: TObject);
begin
    Ferma_M.Tferma_form(Main_Form.ActiveMDIChild).N18.Click;
end;
procedure TMain_Form.FermNextBtnClick(Sender: TObject);
begin
    Ferma_M.Tferma_form(Main_Form.ActiveMDIChild).N19.Click;
end;
procedure TMain_Form.TokPrevBtnClick(Sender: TObject);
begin
    Tok_M.TTok_form(Main_Form.ActiveMDIChild).N18.Click;
end;
procedure TMain_Form.N20Click(Sender: TObject);
var
    i,k,nc,num_ferm: integer;
begin
    for I := MDIChildCount-1 downto 0 do
    begin
        Main_Form.MDICHildren[I].Close;
    end;
end;

```



```

end;
Main_Form.MainMenu1.Items[1].Enabled :=false;
Main_Form.Ferma_Graph_Enter_Panel.Visible:=false;
Main_Form.Plast_Graph_Enter_Panel.Visible:=false;
Main_Form.TOK_Graph_Enter_Panel.Visible:=false;
Ferma_Fd_Form.first_show_FD_form :=true;
Plast_Fd_Form.first_show_FD_form :=true;
TOK_Fd_Form.first_show_FD_form :=true;
Main_Form.Ferma_Panel.Visible:=false;
Main_Form.Plast_Panel.Visible:=false;
Main_Form.TOK_Panel.Visible:=false;
Ferma_Fd_Form.Close;
Plast_Fd_Form.Close;
Tok_Fd_Form.Close;
Main_Form.FermaGraphButton.Down :=false;
Main_Form.FermaNumberButton.Down :=false;
Main_Form.PlastGraphButton.Down :=false;
Main_Form.PlastNumberButton.Down :=false;
Main_Form.TOKGraphButton.Down :=false;
Main_Form.TOKNumberButton.Down :=false;
Main_Form.SimpleResult_TB.ImageIndex:=8;
Main_Form.TOK_OK_PMI.Checked :=false;
Main_Form.TOK_NO_PMI.Checked :=true;
Main_Form.F_Save_TBtn.Enabled:=false;
Main_Form.Caption:='Ферма';
Main_Form.Main_Panel.Visible:=true;

end;
procedure TMain_Form.Sort(Sender: TObject);
var
i,j: integer;
s,ss: string;
t: TStringList;
found: integer;
begin
t := TStringList.Create;
for i:=1 to Main_form.MDICHildCount do begin
t.Add(Main_form.MDICHildren[i-1].Caption);
end;
t.Sort;
found:=0;
for j:=0 to t.Count-1 do begin
for i:=1 to Main_form.MDICHildCount do begin
if (found=0) then begin
if (Main_form.MDICHildren[i-1].Caption=t.Strings[j]) then begin
Main_form.MDICHildren[i-1].SetFocus;
found:=1;
end;
end;
end;
sorted:=true;
end;
end;

```

end;
end.