

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Приладобудівний факультет

Кафедра приладів і систем орієнтації і навігації

«До захисту допущено»

Завідувач кафедри

_____ Бурау Н. І.

«__» _____ 20__ р.

Дипломний проект

на здобуття ступеня бакалавра

**за спеціальністю 151 Автоматизація та комп'ютерно-інтегровані
технології**

**на тему: «Кінематичний аналіз плоского важільного механізму
програмними засобами(робота).»**

Виконав:

студент III курсу, групи ПГ-пб1

Ступак Олександр Юрійович _____

Керівник:

к. т. н. Цибульник С. О. _____

Консультант з: _____

Рецензент:

доц. Маркіна О. М. _____

Засвідчую, що у цьому дипломному
проекті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Київ – 2019

Анотація

У наш час завдання кінематичного аналізу механізмів найчастіше вирішуються графічними або аналітичними методами. З розвитком мов програмування та можливостей відповідних середовищ розробки стає можливою реалізація кінематичного аналізу плоских важільних механізмів програмними методами. Запропонована реалізація являється комбінованим графоаналітичним методом, оскільки аналітичне рішення засноване на графічних побудовах та рішенні відповідних геометричних задач. За допомогою мови програмування Processing реалізовано програмне забезпечення для кінематичного аналізу простих важільних механізмів.

Кінематичний аналіз, важільний механізм, мова програмування Processing.

Ключові слова: Processing, мова моделювання, креслення, система автоматизованого проектування (САПР), прикладна механіка, план положень, план швидкостей, план прискорень.

Реферат

В наше время задача кинематического анализа механизмов чаще всего решаются графическими или аналитическими методами. По развитию языков программирования и возможности соответствующих сред разработки становится возможна реализация кинематического анализа плоских рычажных механизмов программными методами. Предложена реализация является комбинированном графоаналитическим методом, поскольку аналитическое решение основано на графических построениях и решении соответствующих геометрических задач. С помощью языка программирования Processing реализовано программное обеспечение для кинематического анализа простых рычажных механизмов.

Кинематический анализ, рычажный механизм, язык программирования Processing.

Ключевые слова: Processing, язык моделирования, чертежи, система автоматизированного проектирования (САПР), прикладная механика, план положений, план скоростей, план ускорений.

Abstract

Nowadays, the problem of kinematic analysis of mechanisms is most often solved by graphical or analytical methods. According to the development of programming languages and the possibility of corresponding development environments, it becomes possible to implement the kinematic analysis of flat lever mechanisms using software methods. The proposed implementation is a combined graphoanalytical method, since the analytical solution is based on graphical constructions and the solution of the corresponding geometric problems. Using the programming language Processing implemented software for the kinematic analysis of simple lever mechanisms.

Kinematic analysis, lever mechanism, programming language Processing.

Keywords: Processing, modeling language, drawings, computer-aided design (CAD), applied mechanics, position plan, velocity plan, acceleration plan.

Зміст

Вступ	7
1. Проектно – описова частина.....	9
1.1 Історія розвитку інформаційних технологій.....	9
1.2 Теоретичні та практичні складові прикладної механіки.....	11
1.3 Використання комп'ютера для створення креслення механізму	11
1.4 Система автоматизованого проектування	13
1.5 Мова графічного моделювання (Processing).....	14
1.5.1 Історія мови програмування	16
1.5.2 Особливості	16
1.5.3 Переваги.....	17
1.5.4 Недоліки.....	17
2. Кінематичний аналіз плоских важільних механізмів.....	17
2.1 Цілі кінематичного дослідження механізмів	18
2.2 Побудова планів положень	18
2.3 Побудова планів швидкостей	22
2.4 Методи кінематичного аналізу	30
2.4.1 Графічний	30
2.4.2 Аналітичний	30
3. Побудова 12 положень механізму в програмі Processing	32
3.1 – побудова плану прискорень.....	38
3.2 – побудова плану швидкостей	48
3.3 – анімація руху механізму.....	57
4 Побудова 12 положень механізму в SolidWorks.....	64
Висновок.....	65
Список літератури	67

Вступ

З розвитком мов програмування та можливостей відповідних середовищ розробки стає можливою реалізація кінематичного аналізу плоских важільних механізмів програмними методами. Дана реалізація являється комбінованим графоаналітичним методом, оскільки аналітичне рішення засноване на графічних побудовах та рішенні відповідних геометричних задач.

У даній роботі розглянута можливість застосування сучасних комп'ютерно-інтегрованих технологій і мов програмування для побудови планів положень, швидкостей і прискорень плоского важільного механізму. Реалізація побудови названих планів проходила за допомогою мови програмування Processing.

Мови програмування, як і людські мови, об'єднуються в групи споріднених мов. Processing – це діалект мови програмування під назвою Java [2]. Він має майже той же синтаксис, але доповнений спеціальними командами для роботи з графікою і зовнішніми пристроями. Processing містить в собі особливості багатьох мов програмування і тому може послужити хорошим введнням в програмування на інших мовах з використанням інших інструментів розробки.

Processing розроблявся досить довго: з серпня 2002 по квітень 2005 він перебував у стадії альфа-версії, а потім розповсюджувався в стадії бета-версії до листопада 2008. Протягом цього часу він постійно використовувався для навчання та створення програм тисячами людей по всьому світу [1]. Протягом цього часу мова, середовище розробки та спосіб подачі матеріалу безперервно удосконалювалися. 29 листопада 2008 вийшла версія 1.0. Це була перша стабільна версія мови.

Як і будь-яке програмне забезпечення Processing складається з великої кількості компонентів, які працюють разом.

Processing може бути використаний як для простих виробів, так і для докладного дослідження. Програма на Processing може становити від одного до декількох тисяч рядків коду, тому завжди можна поліпшити і розширити її

функціонал. Існує більше 100 бібліотек розширення, що дозволяють застосовувати Processing для обробки звуку, досліджень в області машинного зору і технічних розрахунків. Графічні об'єкти Processing споріднені системі PostScript, що послужила основою для формату PDF і OpenGL (графічна бібліотека для написання додатків, що використовують 3D-графіку). Саме широкий функціонал Processing став вирішальним фактором у виборі мови програмування для вирішення задач кінематичного аналізу механізмів програмними засобами.

1.1 Історія розвитку інформаційних технологій

Історія розвитку інформаційних технологій досягла апогею після створення ПЕОМ в 1981 році. Основні інформаційні технології, які використовуються, представлені на рисунку 1 [1].

Інформаційні технології (ІТ) - процеси, методи пошуку, збору, зберігання, обробки, надання, поширення інформації і способи здійснення таких процесів і методів [1].

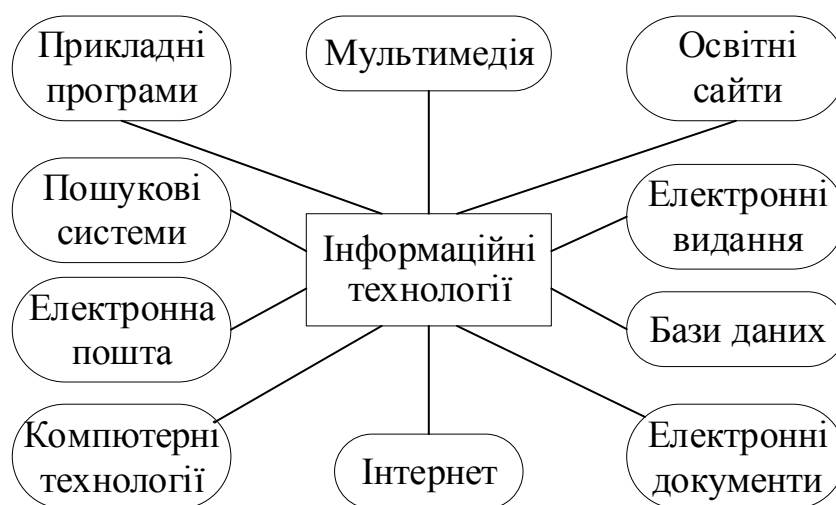


Рисунок 1 - Інформаційні технології

Бурхливий розвиток технологій в останні півстоліття суттєво вплинув практично на всі сторони людського життя. Осмислення цього впливу знайшло відображення в спробах визначити сучасне суспільство як інформаційне. Значний вплив комп'ютерних технологій змінив обличчя, як багатьох окремих галузей сучасної науки, так і обличчя науки в цілому.

Перший комп'ютер (ENIAC) було створено інженерами Пенсільванського університету для розрахунку артилерійських таблиць стрільби, по суті, для вирішення механічних задач балістики. Уже перші тестові розрахунки принесли цікавий результат – 30-та секундна траєкторія польоту снаряда розраховувалася за 20 секунд. Це була перша вказівка на можливість створення з допомогою комп'ютера систем керування швидкоплинними процесами в реальному масштабі часу [2].

Одержані при аналізі кількісних співвідношень результати в оцінці поведінки нелінійної механічної системи вперше проілюстрували можливості комп'ютера, як інструмента одержання принципово нових фундаментальних знань. Ці результати стали основою для розвитку нових методів нелінійної динаміки. Практично в цей самий час використання комп'ютера стало стимулом для розвинення потужного методу дослідження в багатьох галузях науки – методу Монте-Карло [2].

Широкі можливості одержання принципово нових результатів в механіці з допомогою комп'ютерів відмічалися вже в узагальнюючій статті про роботу третього Всесоюзного з'їзду по теоретичній та прикладній механіці в 1969 році. Уже в цей час на з'їзді було представлено низку доповідей, що чітко свідчили про таку можливість. Автори вказаної статті відмічали, що застосування комп'ютерів в механіці наклало свій відбиток на саму постановку багатьох задач механіки [2].

Наступним важливим кроком у розвитку механіки та інших природничих наук стало формування теорії детермінованого хаосу. Величезна робота по створенню теоретичного базису такої теорії була проведена багатьма вченими, в основному математиками, на початку та в середині ХХ століття. Певною мірою початок цієї теорії можна пов'язувати з роботами А. Пуанкаре. Однак основним стимулом для розвитку цієї теорії стали результати аналізу нелінійних систем, одержані з допомогою комп'ютерів. Особливо відзначають роботу Е. Лоренца, присвячену аналізу результатів поведінки досить простої, на перший погляд, системи з трьома степенями вільності. Система призначалася для формування прогнозу погоди. Практично випадково при проведенні обчислень автор встановив наявність однієї з характерних ознак систем з хаотичною динамікою – надзвичайну чутливість кількісних оцінок параметрів системи до незначних змін в початкових умовах. Подальший аналіз кількісних результатів призвів до формування поняття дивного атратора. Що стосується передбачення погоди, то висновок автора

був категоричним – передбачити погоду більше, ніж на тиждень, принципово неможливо [2].

1.2 Теоретичні та практичні складові прикладної механіки

Теоретична та прикладна механіка, є найважливішими розділами природознавства, які грають також провідну роль в розробці теоретичної бази інженерної справи, залучаючи для вирішення поставлених завдань методи фізичного дослідження, математичного, комп'ютерного аналізу і моделювання. Останні при використанні високопродуктивних ЕОМ дозволяють значно скоротити час і знизити економічні витрати на наукові дослідження. Розвиток інформаційних технологій, зростання обчислювальних ресурсів, поява спеціалізованої апаратури докорінно змінило уявлення про рішення математичних задач, і математичне і комп'ютерне моделювання стало провідним інструментом механіки [3].

Практичні потреби розв'язання складних інженерних проблем механіки та можливості використання комп'ютерів для цього стимулювали розробку широкого спектру програмних комплексів для моделювання задач механіки. Як характерні приклади можна відзначити відомі комерційні (дуже дорогі) комплекси ANSYS, FLUENT. Співдружність математиків та механіків розвиває також комплекси з вільним доступом, які також досить широко використовуються. З допомогою таких програмних комплексів та сучасних потужних суперкомп'ютерів вдається успішно розв'язувати надзвичайно складні задачі. При цьому заміна фізичного експерименту комп'ютерним приносить значний вигреш, як економічний, так і вигреш в часі реалізації проекту [4].

1.3 Використання комп'ютера для створення креслення механізму

Оцінюючи визначні досягнення в використанні комп'ютера при розв'язанні складних задач, слід звертати увагу і на певні проблеми. Накопичений досвід комп'ютерного моделювання свідчить про необхідність обережного підходу до оцінки його результатів. Досить чітко проблема сформульована в назві одного з розділів монографії: Reliability without Truth

(надійність без адекватності). Саме ця обставина стимулює дуже активну роботу по створенню стандартів для тестування результатів комп'ютерного моделювання.

Лідируючі позиції тут належать американцям. Практично всі державні установи, що фінансують наукові дослідження створили свої нормативні документи, що регламентують таке тестування. Декілька стандартів розроблено і Американським товариством інженерів-механіків. Один із них (ASME V&V 20 2009), визнаний як державний стандарт США, на 85 сторінках регламентує процедуру оцінки результатів моделювання в механіці рідини та в дослідженні процесів теплопередачі. [4]

Розвиток комп'ютерної техніки зараз дійсно суттєво впливає на саму постановку задач механіки. Побіжний перегляд каталогу бібліотеки одного з провідних західних університетів вказує на видання більше 15 наукових журналів механічного профілю, назва яких починається зі слів COMPUTER, COMPUTING, COMPUTATIONAL. Постійно зростає потужність обчислювальних машин. [4]

Проектування будь-якого механізму завжди супроводжується створенням проектного документа – креслення. Якщо раніше для цієї мети потрібно було використовувати «ватмани», олівці і лінійки, то сьогодні без комп'ютерної програми не обійтись. Все тому, що вони мінімізують ймовірність помилок, допущених при розрахунках або взагалі, усувають їх автоматично. Таким чином, можна отримати якісний проект механізму, який залишиться тільки втілити в життя.

Основними недоліками ручного креслення є складність побудови – дотримання розмірів ліній, градусів кутів, паралельність ліній і т.д. Все це забирає багато часу і вимагає досить багато зусиль.

Завдяки використанню автоматизованого проектування на комп'ютері, значно збільшується швидкість створення кінцевого продукту. Виконання креслень в програмах дозволяє підвищити не тільки швидкість, але і якість самої роботи.

Якщо раніше процес виправлення, доробки та розмноження креслень з різними параметрами був цілою проблемою і забирав багато часу і матеріальних ресурсів, то зараз робиться за допомогою декількох клацань мишею. Завдяки комп'ютерному кресленню і Інтернету тепер креслення по електронній пошті передаються за лічені секунди. У таких програмах можна організувати колективну роботу, коли люди працюють з одним файлом і всім відразу видно зміни і правки не залежно від того, де територіально знаходяться і на якій відстані один від одного учасники колективної роботи.

В даний час існує чимало платних і безкоштовних програм, за допомогою яких можна проектувати різноманітні і в тому числі промислові механізми навіть в 3D-форматі. Це досить зручно, тому що крім двомірного креслення можна побачити реальну модель проекту [5].

1.4 Система автоматизованого проектування

Система автоматизованого проектування (САП або САПР) або автоматизована система проектування (АСП) — автоматизована система, призначена для автоматизації технологічного процесу проектування виробу, результатом якого є комплект проектно-конструкторської документації, достатньої для виготовлення та подальшої експлуатації об'єкта проектування. Реалізується на базі спеціального програмного забезпечення, автоматизованих банків даних, широкого набору периферійних пристроїв [6].

САПР виконує такі функції:

1. конструкторська частина — розробка повного комплексу конструкторської документації;
2. технологічна частина — розрахунок і проектування технологічних схем, технологічного оснащення, транспорту;
3. архітектурно-будівельна частина — розрахунок і проектування металевих і залізобетонних конструкцій;
4. санітарно-технічні системи — проектування теплопостачання, опалення і вентиляції виробничих і адміністративних корпусів, а також водопостачання і каналізації;

5. електротехнічні системи — розрахунок і проектування електропостачання, електросилового устаткування, світлотехнічної частини проектів, телемеханізації електропостачання;

6. гідротехнічні спорудження — розрахунок і проектування напірного і безнапірного гідротранспорту відвальних хвостів, стійкості укосів хвостосховищ;

7. системи автоматизації — розробка схем зовнішніх з'єднань, електричних і трубних проводок щитів автоматики;

8. кошторисна частина — складання локальних і зведених кошторисів, відомостей матеріалів, специфікацій, комплектація обладнання [6].

1.5 Мова графічного моделювання(Processing)

Мова моделювання — це будь-яка штучна мова, котра може використовуватися для вираження інформації (даних) або знань чи систем у структурі, й яка визначається послідовним набором правил. Правила застосовуються задля інтерпретації значення компонентів у структурі. В області інформатики та суміжних галузях керування даними або процесами, мови моделювання дозволяють розробникам програмного забезпечення, системним аналітикам чи системним архітекторам, визначати вимоги до організації або системи програмного забезпечення, а також до її структур та внутрішніх процесів, з більш високим рівнем абстракції [7].

Переваги мов моделювання.

1. Мови графічного моделювання використовують діаграмну техніку з іменованими символами, які представляють поняття та лінії, що зв'язують символи та надають співвідношення і різні інші графічні позначення, для подання обмежень.

2. Мови текстового моделювання, можуть використовувати стандартизовані ключові слова, супроводжувані параметрами або термінами та фразами на природній мові, щоби зробити вирази, які визначаються комп'ютером [7].

Прикладом мови графічного моделювання та відповідної мови текстового моделювання, є Processing.

Таким чином для виконання завдання найкраще підходить мова модулювання Processing.

Що таке Processing



Рисунок 2 - Логотип програми Processing

Processing – відкрита мова програмування, заснована на Java. Являє собою легкий і швидкий інструментарій для людей, які хочуть програмувати зображення, анімацію та інтерфейси.

Використовується студентами, художниками, дизайнерами, дослідниками і любителями, для вивчення, прототипування і виробництва. Вона створена для вивчення основ комп'ютерного програмування у візуальному контексті і служить альбомним програмним забезпеченням (мається на увазі те, що кожен * .pde файл візуальної оболонки Processing'а є окремим зображенням або анімацією, і т. д.) і професійним виробничим інструментом.

Processing – це відкритий проект ініційований Бенжаміном Фраєм і Кейсі Різом. Він народився з ідей, вивчених в The Aesthetics and Computation Group в MIT Media Lab. Мова Processing та IDE були попередниками численних інших проектів, зокрема Arduino, Wiring та p5.js.

1.5.1 Історія мови програмування

Проект був ініційований у 2001 році Кейсі Реасом і Бен Фрі, як колишньою групою естетики та обчислень у медичній лабораторії МІТ. У 2012 році разом з Даніелем Шіффманом вони почали працювати з Фондом обробки даних, який приєднався як третій керівник проекту. Йоханна Хедва приєдналася до Фонду в 2014 році як директор з адвокатури [8].

Спочатку Processing мав URL на `proce55ing.net`, тому що домен Processing був зайнятий. Зрештою Reas і Fry придбали домен `processing.org`. Хоча назва мала поєднання букв і цифр, вона все ще була виражена частина Processing. Вони не віддають перевагу середовищу, що називається `Proce55ing`. Незважаючи на зміну доменного імені, Processing досі використовує термін `p5` іноді як скорочене ім'я (`p5` конкретно, а не `p55`), наприклад `p5.js` є посиланням на це [8].

1.5.2 Особливості

Processing включає в себе етюдник, мінімальну альтернативу інтегрованому середовищу розробки (IDE) для організації проектів.

Кожен ескіз обробки є фактично підкласом класу `Applet Java` (колишній підклас вбудованого аплету Java), який реалізує більшість функцій мови Processing.

При програмуванні в Processing всі додаткові визначені класи будуть розглядатися як внутрішні класи, коли код перетворюється на чисту Java перед компіляцією. Це означає, що використання статичних змінних і методів у класах заборонене, окрім випадків, коли обробка явно вказана в коді в чистому режимі Java.

Processing також дозволяє користувачам створювати власні класи в рамках ескізу `Applet`. Це дозволяє використовувати складні типи даних, які можуть включати будь-яку кількість аргументів і уникає обмежень виключно з використанням стандартних типів даних, таких як: `int` (ціле), `char` (символ), `float` (реальне число) і колір (`RGB`, `RGBA`, `hex`) [8].

1.5.3 Переваги

1. Інтерактивність – Це простий спосіб отримати і використовувати введення за допомогою миші і клавіатури і визначити обробники подій.
2. Графіка – Processing має простий, але потужний графічний API. 2D дійсно простий, і він має потужні функції, які можна використовувати без будь-яких шаблонів, таких як 3D-рендерінг і матричний стек.
3. Інженерний потік – як вже було зазначено вище, короткий цикл ітерації, ясний синтаксис і простий API роблять Processing легким, і швидким в роботі.
4. Multiplatform – з коробки, можна створювати власні програми для Windows, Mac і Linux, рідні програми для Android і веб-аплети. Через Processing.js ви також можете запускати ескізи в веб-браузерах [9].

1.5.4 Недоліки

1. Великі, складні програми (вони, як правило, стають трохи громіздкими, оскільки неможливо організувати класи у структурі папок.)
2. Допомога програмістам (Немає відладчика за межами коду. Помилки препроцесора часто неспецифічні і можуть залишити вас шукати помилки синтаксису в декількох файлах.)
3. Аудіо (немає жодної підтримки, але доступні бібліотеки) Processing тепер включає відмінну бібліотеку Minim, але ви повинні явно імпортувати її. Звук як і раніше виглядає як запізнілий в документації та навчальних посібниках [9].

2. Кінематичний аналіз плоских важільних механізмів

Завданням кінематичного аналізу є знаходження траєкторій, швидкостей і прискорень точок ланок механізму, кутових швидкостей і кутових прискорень ланок механізму. Вихідні дані: кінематична схема механізму (з розмірами ланок) і закон руху початкової ланки (зазвичай – кривошипа). Методи кінематичного аналізу: метод планів, метод кінематичних діаграм, аналітичний метод (метод замкнутого векторного контуру).

На стадії усталеного руху досить зробити кінематичний аналіз в межах одного циклу (періоду зміни узагальненої координати початкової ланки), як правило, це один або два обороти кривошипа.

2.1 Цілі кінематичного дослідження механізмів

Кінематичне дослідження механізму переслідує три основні цілі:

1. Вивчення положень механізму. Вивчення переміщень точок і кутів повороту ланок. Визначення траєкторій руху точок. Результати досліджень по даному пункту використовуються при визначенні габаритних розмірів механізму, визначенні форми та розмірів внутрішньо корпусного простору, визначення швидкостей, прискорень, вирішенні задач динамічного синтезу та аналізу.

2. Визначення швидкостей точок і кутових швидкостей ланок.

3. Знайдені лінійні швидкості можуть бути використані, в тому числі, для визначення потужностей сил і наведеної сили; кутові швидкості – для визначення нормальних прискорень точок, потужностей моментів сил і інших цілей.

4. Визначення прискорень точок і кутових прискорень ланок. Прискорення в задачах динаміки дозволяють виконати силовий розрахунок механізму, тобто розрахунок рушійної сили і навантаженості кінетичних пар за методом кінетостатики (з використанням принципу Даламбера). У кулачкових механізмах по прискорень судять про динамічних навантаженнях, що виникають в процесі роботи механізму [10].

2.2 Побудова планів положень

План механізму – це графічне зображення в масштабі взаємного розташування ланок при заданому значенні узагальненої координати.

Побудова планів положень є обов'язковим етапом при вирішенні задачі кінематичного аналізу графічним методом. Тільки плани положень дозволяють визначити траєкторії окремих точок механізму. Тільки з їх допомогою можна будувати плани швидкостей і плани прискорень, визначити

кутові швидкості і кутові прискорення ланок, виконувати динамічні розрахунки.

Побудова планів положень починають із зображення елементів стійки, тобто шарнірно-нерухомих опор і направляючих. Далі послідовно зображують провідні ланки в заданих положеннях і структурні групи ланок. Положення рухомих характерних точок визначаються за допомогою методу зарубок. Якщо провідна ланка здійснює рівномірний обертальний рух, то траєкторією руху однієї з його характерних точок є окружність. Дану траєкторію (окружність) ділять на рівні частини: 12, 24, 36, 48 і т. Д. Кожній отриманій точці присвоюється відповідний номер.

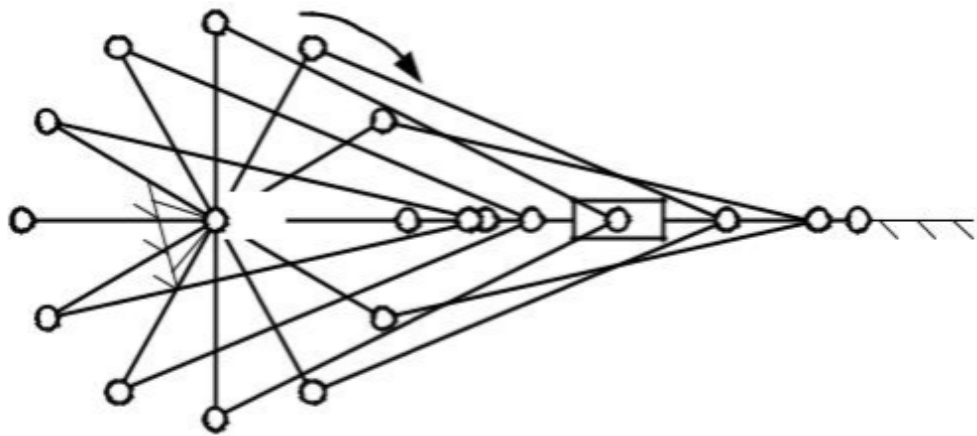


Рисунок 3 - 12 положень механізму

Для кращого розуміння питання про побудову планів положень можна розглянути систематично, виділивши вихідні дані, мету і порядок вирішення завдання.

Постановка задачі:

Дано:

1. Кінематична схема механізму (принципова схема, побудована в масштабі).
2. Положення вхідної ланки або кілька її можливих положень.

Визначити:

1. Зображення належать стійці шарнірів, направляючих і тому подібних.
2. Визначення крайніх положень механізму.

Крайнім положенням механізму прийнято вважати такий стан, при якому хоча б одна ланка механізму займає крайнє положення. При цьому під крайнім положенням ланки розуміють то його положення, з якого воно може рухатися тільки в одному напрямку. Шлях ланки, пройдений від одного крайнього положення до іншого і вимірний в лінійних або кутових одиницях виміру, називають ходом ланки.

Крайні положення механізмів, вхідним ланкою яких є кривошип, знаходяться за відомими правилами:

кривошипно-коромислових (Рисунок. 4) і кривошипно-повзунових, механізм в крайніх положеннях кривошип OA і шатун AB знаходяться на одній прямій OA і CB .

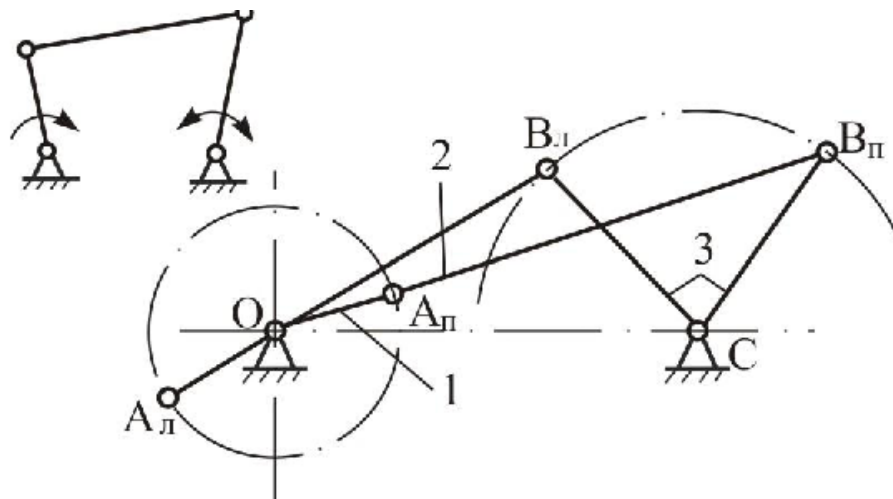


Рисунок 4 - Кривошип-коромисловий

1. точки O і C ;
2. окружність $r = OA$;
3. окружність $R = CB$;
4. дуги $OB_L = AB - OA$;
5. $OB_P = AB + OA$
6. точки B_L, B_P, A_L, A_P

Кінематична схема двохкривошипного кулісного механізму (Рисунок. 5)

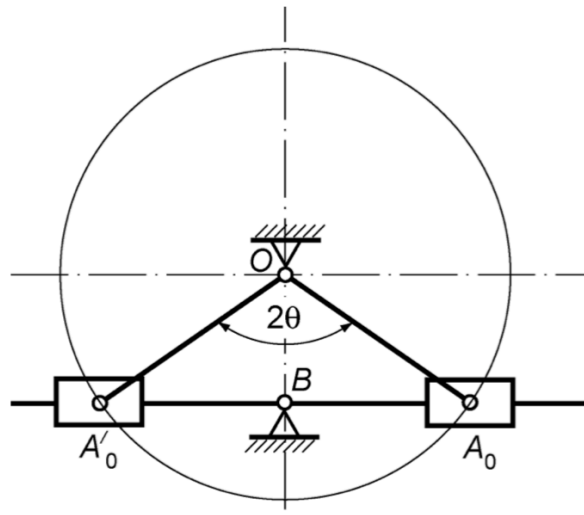


Рисунок 5 - Кінематична схема двохкривошипного кулісного механізму.

Умова прокручування кривошипа OA і куліси BA полягає в тому, щоб довжина стояка l_{OB} була меншою за довжину кривошипа l_{OA} , тобто щоб $l_{OA} > l_{OB}$. У крайніх положеннях механізму куліса BA займає горизонтальне положення.

Відповідно ведуча точка A кривошипа займає положення A_0 і A_0' . Нехай кривошип OA обертається проти годинникової стрілки. Тоді при обертанні його з положення OA_0 в положення OA_0' він повернеться на кут $\varphi_{p.x}$, а з положення OA_0' в положення OA_0 - на кут $\varphi_{x.x}$. Позначимо $\varphi_{x.x} = 2\theta$. Тоді $\varphi_{p.x} = 360^\circ - 2\theta$. Підставимо вирази цих кутів в рівняння :

$$k = \frac{360^\circ - 2\theta}{2\theta} \quad (2.1)$$

Звідки отримуємо значення кута θ :

$$\theta = \frac{180^\circ}{k+1} \quad (2.2)$$

Довжину кривошипа OA визначаємо з трикутника OBA_0 :

$$l_{OA} = \frac{l_{OB}}{\cos\theta} \quad (2.3)$$

2.3 Побудова планів швидкостей

Визначення швидкостей точок (побудова планів швидкостей) і кутових швидкостей ланок механізму:

План швидкостей – це пучок векторів, виконаний в певному масштабному коефіцієнті, промені якого зображують вектори лінійних швидкостей характерних точок механізму, а відрізки, що з'єднують їх вершини, відповідають векторам відносних швидкостей.

Масштабний коефіцієнт плану швидкостей, м / (с * мм), розраховується за формулою:

$$\mu_v = \frac{V_{O_4}}{|pa|} \quad (2.4)$$

де $|pa|$ - довільний відрізок, мм.

Алгоритм побудови планів швидкостей, починаючи з постановки задачі.

Постановка задачі:

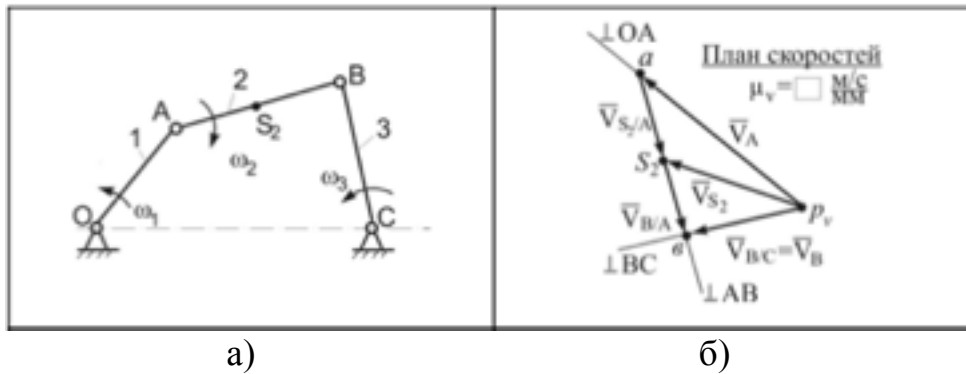
Дано:

1. План положення механізму (кінематична схема механізму в заданому положенні).
2. Швидкість (кутова швидкість) вхідного ланки.

Визначити:

1. Швидкості характерних точок механізму (кінцеві точки ланок і точки з'єднання ланок).
2. Швидкості будь-яких інших точок механізму.
3. Кутові швидкості ланок.

Побудова планів швидкостей для складних механізмів є компіляцією операцій, які виконуються для простих.



а)
 б)
 Рисунок 6 – а) Кривошип коромисловий
 б) План швидкостей

Дано:

Кривошип коромисловий (Рисунок.6.а) в заданому положенні; ланка 1 – вхідний; ω_1 - кутова швидкість ланки 1; $AS_2 = 0,5 \cdot AB$.

Визначити:

Швидкості точок А, В, S2 графічним шляхом (побудовою плану швидкостей).

Рішення

Визначення швидкостей, як і будь-яка інша задача кінематичного аналізу, виконується поетапно, в послідовності, визначеною формулою будови механізму.

У нашому випадку формула будови має вигляд: $I(0,1) \rightarrow II(2,3)$

З формули впливає порядок можливих дій:

1. визначення і зображення на кресленні швидкостей точок первинного механізму;
2. визначення і зображення швидкостей точок і кутових швидкостей ланок структурної групи $II(2,3)$.

Слідуючи цим порядком, спочатку зображують на кресленні швидкості характерних точок механізму (кінцевих точок ланок і точок рухомих сполук ланок). Отриманий «базовий» план швидкостей використовують для визначення абсолютних значень швидкостей самих характерних точок механізму і кутових швидкостей ланок. На другому етапі побудов знаходять і зображують на плані швидкості будь-яких інших точок.

Реалізуємо даний алгоритм для побудови плану і визначення швидкостей точок і ланок нашого механізму.

1. Визначимо і покажемо на плані (Рисунок.6.б) швидкість точки А.

Точка А є характерною точкою механізму. Вона належить ланці 1, яка обертається навколо нерухомого центру О з кутовою швидкістю ω_1 .

Тому:

1. напрям швидкості $\vec{V}_A \perp OA$ в напрямі ω_1
2. Для графічних побудов виберемо масштабний коефіцієнт плану швидкостей:

$$\mu_v = \frac{V_{AO}}{pa} \quad (2.5)$$

Впливає, що швидкість \vec{V}_A на плані буде зображена відрізком довжиною $p_v a$. В поле креслення виберемо довільну точку p_v - полюс плану швидкостей. Зобразимо вектор $\vec{V}_A = p_v a$, відклавши його від точки p_v перпендикулярно ОА в напрямку кутової швидкості кривошипа.

Визначимо і покажемо на кресленні швидкість точки В.

Точка В також є характерною точкою механізму. Це загальна точка для ланок 2 і 3. Для визначення швидкості точки В слід графічно вирішити систему векторних рівнянь:

$$\begin{cases} \vec{V}_B = \vec{V}_A + \vec{V}_{B/A} \\ \vec{V}_B = \vec{V}_C + \vec{V}_{B/C} \end{cases} ; \vec{V}_C = 0; \quad \begin{matrix} \vec{V}_{B/A} \perp AB \\ \vec{V}_{B/C} \perp BC \end{matrix} \quad (2.6)$$

В системі двома рисами підкреслені вектори величина і напрямок яких в даний момент є відомими. Однією рисою підкреслені вектори, у яких відомо тільки їх напрямок. Якщо відомі вектори \vec{V}_A і \vec{V}_C відкласти з однієї точки (наприклад, з полюса p_v) і домалювати до них напрямку $\vec{V}_{B/A}$ і $\vec{V}_{B/C}$ то завдання по визначенню невідомого вектора \vec{V}_B можна вирішити графічно. Тоді точка P_v буде початком, а точка «в» (отримана на перетині напрямків $\vec{V}_{B/A}$ і $\vec{V}_{B/C}$) –

кінцем вектора $\overline{p_v b} = \overline{V_B}$ задовольняє одночасно обом рівнянням системи. Таким чином, величини і напрямки $\overline{V_{B/A}}$ і $\overline{V_{B/C}}$ також стають визначеними. З урахування $\overline{V_C} = 0$, маємо $\overline{V_B} = \overline{V_{B/C}}$.

Числове значення величини V_B знайдемо з плану, вимірявши довжину відрізка $p_v b$:

$$V_B = p_v b * m_v \quad (2.7)$$

Базовий план швидкостей побудований.

Тепер потрібно визначити швидкості інших точок. Такою точкою за умовою задачі є точка

S_2 - центр мас ланки 2.

Визначимо швидкість точки S_2

Швидкість точки S_2 знаходимо за аналогічною векторною формулою. В цьому випадку система рівнянь не знадобиться, тому що вектор $\overline{V_A}$ визначений раніше, а величина і направленість $\overline{V_{S_2/A}}$ легко визначається (на підставі відомого $\overline{V_{N/A}}$).

$$\overline{V_{S_2}} = \overline{V_A} + \overline{V_{S_2/A}} \quad (2.8)$$

$$p_v S_2 = \overline{V_{S_2}} \quad (2.9)$$

Визначимо кутові швидкості ланок 2 і 3:

Кутові швидкості ω_2 і ω_3 знайдемо за формулами:

$$\omega_2 = \frac{V_{B/A}}{I_{A/B}} \quad \omega_3 = \frac{V_B}{I_{B/C}} \quad (2.10)$$

Напрямки ω_2 і ω_3 зображуємо на рисунок. 8.1 відповідно до напрямів $\overline{V_{B/A}}$ і $\overline{V_{B/C}}$ на плані швидкостей.

З визначення плану швидкостей випливають його властивості:

1) всі вектори, що становлять план, є векторами швидкостей характерних точок механізму;

2) всі вектори швидкостей, що виходять з полюса плану (точки p), є векторами лінійних швидкостей характерних точок механізму;

3) всі вектори швидкостей, що не проходять через полюс плану (точку p), є векторами відносних швидкостей характерних точок механізму;

4) швидкості характерних точок механізму, рівні нулю, зображуються точковими векторами, що збігаються з полюсом плану швидкостей (точкою p) [11].

Визначення прискорень точок (побудова планів прискорень) і кутових прискорень ланок механізму.

План прискорень – це пучок векторів, виконаний в певному масштабному коефіцієнті, промені якого зображують вектори абсолютних прискорень характерних точок механізму, а відрізки, що з'єднують їх вершини, відповідають векторам відносних прискорень.

Масштабний коефіцієнт плану прискорень, $m / (c^2 \cdot \text{мм})$, обчислюють за формулою

$$\mu_a = \frac{a_{OA}^{\pi}}{|\pi a|} \quad (2.11)$$

де - $|\pi a|$ довільний відрізок, мм.

Кутове прискорення – це відношення тангенціального (дотичного) прискорення ланки механізму до дійсної довжини цієї ланки.

Напрямок дії кутового прискорення ланок, що здійснюють обертальні або складні рухи, вказує вектор тангенціального прискорення характерних точок відповідної ланки, перенесений з плану прискорень в однойменну точку, що належить цієї ланки на схемі механізму. При цьому розривається зв'язок розглянутого ланки з іншими ланками, а до вільної характерній точці прикладається шарнірно-нерухома опора. У цьому випадку дана точка стає умовно нерухомою, а однойменна точка спільно з ланкою під дією вектора тангенціального прискорення отримує можливість здійснювати обертальний рух навколо умовно нерухомої точки в напрямку дії цього вектора. Отриманий напрям обертального руху розглянутого ланки є напрямком дії кутового

прискорення цієї ланки. Кутове прискорення ланок механізмів, що здійснюють поступальні рухи, дорівнює нулю.

Теорема подібності

Моделі плоских важільних механізмів можуть містити характерні точки, які є центрами кінематичних пар, які утворені ланками, які не мають зв'язків з елементами стійки. Визначення швидкостей і прискорень подібних точок здійснюється по теоремі подоби, яка формулюється так: відрізки, що з'єднують точки на схемі (плані положень) механізму, і відрізки, що з'єднують однойменні точки на планах швидкостей або прискорень, утворюють подібні фігури.

Якщо порядок букв при обході по контуру в обраному напрямку однаковий, то подібні фігури до того ж і подібно розташовані. Фігура, отримана на плані прискорень, буде повернута відносно результатній фігури схеми (плану положень) механізму на деякий кут в напрямку обертання ведучого ланки.

Згідно з формулюванням теореми подібності, характерна точка, являється центром кінематичної пари, утвореної ланками механізму, що не мають зв'язків зі стійкою, лежить на схемі механізму на деякій ланці, отже, однойменна точка як на плані швидкостей, так і на плані прискорень розташована на відрізку, що зображає ця ланка в складі обох планів. Склавши пропорцію, що характеризує відношення дійсних довжин ланок і відрізків, які відповідають цим параметрам в складі планів, знайдемо довжину відрізка, що визначає положення даної точки як на плані швидкостей, так і на плані прискорень. Відклавши довжину отриманого відрізка на планах швидкостей і прискорень, установами положення шуканої точки. Поєднавши знайдені точки з полюсами планів, отримаємо відрізки, пропорційні, відповідно, векторах швидкості і прискорення даної характерної точки. Отримані вектори швидкості і прискорення будуть спрямовані від полюсів планів до знайдених точок. Значення швидкості і прискорення даної характерної точки розрахуємо як твір довжини відрізка з відповідного плану на його масштабний коефіцієнт.

Постановка задачі

Дано:

1. План механізму в заданому положенні.
2. Кутова швидкість і кутове прискорення ланки приводу.
3. Кутові швидкості всіх інших ланок механізму.

Визначити:

1. Прискорення характерних точок (точки з'єднання ланок).
2. Прискорення інших точок.
3. Кутові прискорення ланок.

Кривошипно-коромисловий механізм

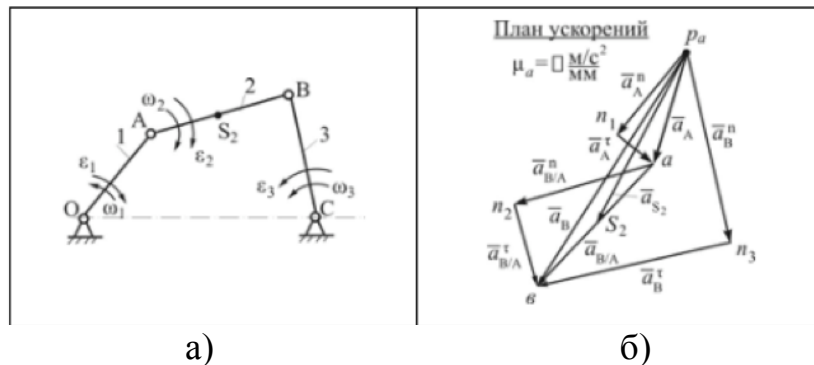


Рисунок 7 – а) Кривошип коромисловий
б) План швидкостей

Рішення

Визначимо прискорення точки A

$$\bar{a}_A = \bar{a}_A^n + \bar{a}_A^t, \quad a_A^n = \omega_1^2 * l_{OA}, \quad a_A^t = \epsilon_1 * l_{OA}, \quad (2.12)$$

$$\text{Напрямок: } \bar{a}_A^n \parallel OA, \quad \bar{a}_A^t \perp OA \quad (2.13)$$

Виберемо масштабний коефіцієнт плану прискорень і побудуємо вектор

\bar{a}_A :

$$\overline{p_a n_1} = \bar{a}_A^n \quad n_1 a = \frac{a_A^t}{m_0} \quad n_1 a = \bar{a}_A^t \rightarrow \bar{a}_A \quad (2.14)$$

Визначимо прискорення точки (точка належить 2-м ланкам):

$$\bar{a}_B = \bar{a}_A + \bar{a}_{B/A} + \bar{a}_{B/A} \quad (2.15)$$

$$\bar{a}_B = \bar{a}_{B/C} + \bar{a}_{B/C} \quad (2.16)$$

$$a_{B/N}^n = \omega_2^2 * I_{AB}, \quad a_{B/C}^n = \omega_3^2 * I_{CB}, \quad a_{B/A}^n \parallel AB, \quad a_{B/A}^n \parallel AB \quad (2.17)$$

Визначимо значення абсолютних і відносних прискорень точок

$$a_B = m_a * p_a b \quad (2.18)$$

Визначимо кутові прискорення ланок

$$e_2 = \frac{a_{B/A}^t}{I_{BA}} \quad e_3 = \frac{a_{B/C}^t}{I_{BC}} \quad (2.19)$$

Напрямы e_2 і e_3 показуємо відповідно до обраного напрямку $\bar{a}_{B/A}^t$ і $\bar{a}_{B/C}^t$ [10].

З визначення плану прискорень випливають його властивості:

- 1) всі вектори, що становлять план, є векторами прискорень характерних точок механізму;
- 2) всі вектори прискорень, що виходять з полюса плану (точки π), являються векторами абсолютних прискорень характерних точок механізму;
- 3) всі вектори прискорень, що не проходять через полюс плану (точку π), є векторами відносних прискорень характерних точок механізму;
- 4) прискорення характерних точок механізму, рівні нулю, зображуються точковими векторами, що збігаються з полюсом плану прискорень (точкою π).

Слідство з властивості . Якщо тангенціальні або радіальні прискорення характерних точок механізму дорівнюють нулю, то вони зображуються точковими векторами, що збігаються на плані прискорень з вершинами векторів (нормальних або Коріолісова) прискорень цих же точок механізму.

Після побудови плану прискорень і визначення значень прискорень всіх характерних точок механізму переходять до визначення значень і напрямів дії кутових прискорень ланок механізму [11].

2.4 Методи кінематичного аналізу

- 1) Графічний (Переваги – наочність, прогнозованість результатів; недоліки – низька точність, трудомісткість).
- 2) Аналітичний (Висока точність, але відсутність наочності, необхідність додаткової роботи по складанню й налагодженню програми).

2.4.1 Графічний

Графічні методи засновані на геометричній побудові траєкторій руху окремих ланок механізму, швидкостей і прискорень їх шарнірів. Отримувані результати дають наочну картину руху ланок механізму і його шарнірів, але необхідні побудови виконуються для кожного конкретного положення механізму. Останнє не дозволяє отримати загальне універсальне рішення [10].

2.4.2 Аналітичний

Аналітичне рішення зручно на добре налагоджених завданнях, що вимагають дослідження вихідних параметрів механізму при різних варіантах вихідних даних або оптимізації цих вихідних даних.

Кінематичному дослідженню механізму, як правило, передують структурний аналіз, в ході якого визначається ступінь рухливості механізму і його клас.

Якщо механізм має ступінь рухливості, рівну одиниці, то положення, швидкості і прискорення точок і ланок механізму – є функції положення, швидкості і прискорення його початкової ланки. За початкову ланку, зазвичай, приймають вхідний ланка механізму, а за узагальнену координату – функцію положення цієї ланки.

Якщо механізм має другий клас, це означає, що він побудований на основі простих діадем. У цьому випадку положення точок і ланок механізму, їх швидкості і прискорення визначаються звичайними методами кінематики.

Аналітичні методи забезпечують високу точність обчислення шуканих параметрів. У даний час дуже часто застосовують саме аналітичні методи. Проте для попередньої оцінки кінематичних параметрів механізму і контролю

аналітичних обчислень використовуються найпростіші геометричні побудови – плани положень, швидкостей і прискорень.

З розвитком мов програмування та можливостей відповідних середовищ розробки стає можливою реалізація кінематичного аналізу плоских важільних механізмів програмними методами. Дана реалізація являється комбінованим графоаналітичним методом, оскільки аналітичне рішення засноване на графічних побудовах та рішенні відповідних геометричних задач [10].

3. Побудова 12 положень механізму в програмі Processing

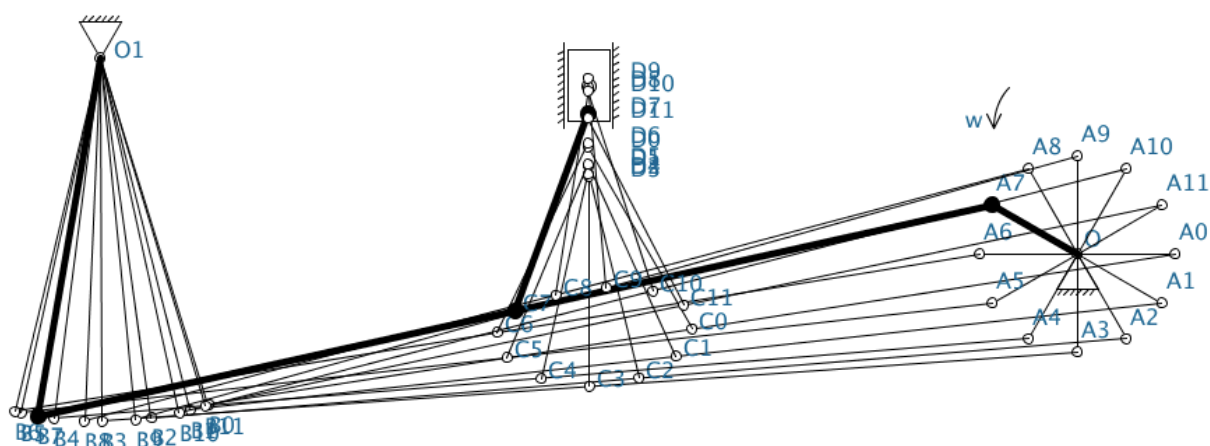


Рисунок 8 – 12 положень механізму

Для знаходження кінцевих координат ланки

```
float[] hinge(float x_start, float y_start, float
angle_rotation, float len){
    float angle_rotation_rad = radians(angle_rotation);
    float x_end = x_start + len * cos(angle_rotation_rad);
    float y_end = y_start + len * sin(angle_rotation_rad);
    float[] end_coordinates = {x_end, y_end};
    return end_coordinates;
}
void paint_shtrix(float leftX, float leftY, float
leftBetween, float lenShtrix, int count, float angel_rotation,
float povorot){
    float x_start_shtrix, y_start_shtrix;
    float angel_rotation_rad = radians(angel_rotation);
    for( int i=1; i<count; i++){
        x_start_shtrix = leftX + leftBetween * i *
cos(angel_rotation_rad);
        y_start_shtrix = leftY + leftBetween * i *
sin(angel_rotation_rad);
        float[] end_shtrix =
hinge(x_start_shtrix,y_start_shtrix,(angel_rotation+povorot),
lenShtrix);
        line(x_start_shtrix,y_start_shtrix,end_shtrix[0],end_shtrix[1]);
    }
}
```

Відмалювання стійки під будь-яким кутом

```
void rack(float x_start, float y_start, float
angel_rotation){
```

Довжина сторони трикутника

```
int len = 30;
```

Кінцеві координати трикутника

```

float[] right_hinge =
hinge(x_start,y_start,(angel_rotation+60),len);
float[] left_hinge =
hinge(x_start,y_start,(angel_rotation+120),len);

```

Відмалювання трикутника

```

line(x_start,y_start,left_hinge[0],left_hinge[1]);
line(x_start,y_start,right_hinge[0],right_hinge[1]);

line(right_hinge[0],right_hinge[1],left_hinge[0],left_hinge[1]);
circle(x_start, y_start, 7);

```

Нижні рисочки під трикутником

```

float len_between_shtrix = len/6.5;
float len_shtrix = 5;
int count = 7;
float povorot = 135;

paint_shtrix(left_hinge[0],left_hinge[1],len_between_shtrix,len_shtrix, count,angel_rotation, povorot);
}

```

```

void slider(float x_center, float y_center, int
angle_rotation ){

```

Відмалювання повзуна під будь-яким кутом

```

float len = 30;

Прямокутник
float
right_downX=x_center+len*cos(radians(angle_rotation+30));
float
right_downY=y_center+len*sin(radians(angle_rotation+30));
float
left_downX=x_center+len*cos(radians(angle_rotation+150));
float
left_downY=y_center+len*sin(radians(angle_rotation+150));
float
left_topX=x_center+len*cos(radians(angle_rotation+210));
float
left_topY=y_center+len*sin(radians(angle_rotation+210));
float
right_topX=x_center+len*cos(radians(angle_rotation+330));
float
right_topY=y_center+len*sin(radians(angle_rotation+330));

```

```

line(right_downX,right_downY,left_downX,left_downY); //
довжина ліва
line(left_downX,left_downY,left_topX,left_topY); //
ширина вгору
line(left_topX,left_topY,right_topX,right_topY); //
довжина права

```



```

        line(right_topX,right_topY,right_downX,right_downY);//
ширина нижня
        circle(x_center,y_center,10);

Смужки поруч з прямокутником
        float len_2 = 35;
        float
right_downX_2=x_center+len_2*cos(radians(angle_rotation+30));
        float
right_downY_2=y_center+len_2*sin(radians(angle_rotation+30));
        float
left_downX_2=x_center+len_2*cos(radians(angle_rotation+150));
        float
left_downY_2=y_center+len_2*sin(radians(angle_rotation+150));
        float
left_topX_2=x_center+len_2*cos(radians(angle_rotation+210));
        float
left_topY_2=y_center+len_2*sin(radians(angle_rotation+210));
        float
right_topX_2=x_center+len_2*cos(radians(angle_rotation+330));
        float
right_topY_2=y_center+35*sin(radians(angle_rotation+330));

line(right_downX_2,right_downY_2,left_downX_2,left_downY_2);
        line(left_topX_2,left_topY_2,right_topX_2,right_topY_2);

Штрихи
        float len_between_shtrix = len_2/6.5;
        float len_shtrix = 5;
        int count = 11;
        int povorot = 60;
        int povorot_1 = -120;

paint_shtrix(left_downX_2,left_downY_2,len_between_shtrix,len_shtrix, count,angle_rotation, povorot);

paint_shtrix(left_topX_2,left_topY_2,len_between_shtrix,len_shtrix, count,angle_rotation, povorot_1);
    }
    float[] tochki_perecechenia_circle(float x1, float y1, float r1, float x2, float y2, float r2 ){

        float d = sqrt( pow(x2-x1,2) + pow(y2-y1,2) );// Відстань між двома центрами окружності
        float a =(pow(r1,2)-pow(r2,2)+pow(d,2))/(2*d);// відрізок до висоти
        float h = sqrt(pow(r1,2)-pow(a,2));// висота
        float xh = x1+a*(x2-x1)/d;// координати початку висоти
        float yh = y1+a*(y2-y1)/d;

        float x3 = xh+h*(y2-y1)/d;// координати точки перетину

```

```

float y3 = yh-h*(x2-x1)/d;

float x4 = xh-h*(y2-y1)/d;
float y4 = yh+h*(x2-x1)/d;
float [] res = {x3,y3,x4,y4};
return res;
}

float[] seredini_otrezka(float x1, float y1, float x2, float
y2) {
float x = (x1+x2)/2;
float y = (y1+y2)/2;
float[] res = {x,y};
return res;
}

void setup() {
size(900,900);
background(255,255,255);

float x_start = 800;
float y_start =440;

rack(x_start,y_start,0);// Головна стійка
rack(100,300,180);// Друга стійка
slider(450,320,90);// Повзун
textSize(16);
fill(0,102,153);
text("0", x_start+5, y_start-5);
fill(255,255,255);
Координати похибки
float eps_for_rack_to_rack = 0.67;
float eps_for_rack_to_slider = 0.38;
float c = 0;

noFill();
arc(780, 350.0, 80, 80, PI, PI+QUARTER_PI);
line(740, 350.0,735, 340.0);
line(740, 350.0,745, 340.0);
fill(0,102,153);
text("w", 720, 350.0);
fill(255,255,255);
for(int angle = 0; angle<360;angle+=30){
if( c == 7 ){
strokeWeight(5);
}else{
strokeWeight(1);
}
}

Координати кінця ланки навколо головного рядка
float[] end_hingle_first =
hinge(x_start,y_start,angle,70);

```

```

        line(x_start, y_start,
end_hingle_first[0],end_hingle_first[1]);// Відмалювання ланки
навколо головної стійки
        circle(end_hingle_first[0],end_hingle_first[1], 7);
        fill(0,102,153);

        text("A"+str(int(c)), end_hingle_first[0]+3,
end_hingle_first[1]-10);
        fill(255,255,255);
Знаходить координати перетину кіл двох стійок
        float[] tochki_perecechenia_circle_rack_to_rack =
tochki_perecechenia_circle(end_hingle_first[0],end_hingle_first[
1],700,100,300,260);// стійка + стійка
Відмалювання ліній
        line(end_hingle_first[0],end_hingle_first[1],
tochki_perecechenia_circle_rack_to_rack[0],tochki_perecechenia_c
ircle_rack_to_rack[1]);

circle(tochki_perecechenia_circle_rack_to_rack[0],tochki_perecec
henia_circle_rack_to_rack[1],7);

Знаходить координати середини відрізка
        float[] koordinati_seredini =
seredini_otrezka(end_hingle_first[0],end_hingle_first[1],
tochki_perecechenia_circle_rack_to_rack[0],tochki_perecechenia_c
ircle_rack_to_rack[1]); // Координати середини для повзуна
Відмалювання по середині кола
circle(koordinati_seredini[0],koordinati_seredini[1],7);

Цикл зміни кута
for(float i=0;i<360;i+=0.3){
        float[] end_hinge_second =
hinge(koordinati_seredini[0],koordinati_seredini[1],i,150);
        float[] end_hingle_third = hinge(100,300,i,260);
Умовний рух повзуна
        if(end_hinge_second[0]>450-
eps_for_rack_to_slider &&
end_hinge_second[0]<450+eps_for_rack_to_slider){
Відкидаємо нижні координати

if(end_hinge_second[1]<koordinati_seredini[1]){

line(koordinati_seredini[0],koordinati_seredini[1],
end_hinge_second[0],end_hinge_second[1]);
        fill(0,102,153);

text("D"+str(int(c)),end_hinge_second[0]+30,
end_hinge_second[1]);
        fill(255,255,255);
        fill(0,102,153);

```

```

text("C"+str(int(c)),koordinati_seredini[0]+5,
koordinati_seredini[1]);
    fill(255,255,255);

circle(end_hinge_second[0],end_hinge_second[1],7);
    }
    }
УМОВНИЙ перетин двох кіл з похибкой
    if( end_hingle_third[0]-
eps_for_rack_to_rack<tochki_perecechenia_circle_rack_to_rack[0]
&&

tochki_perecechenia_circle_rack_to_rack[0]<end_hingle_third[0]+e
ps_for_rack_to_rack &&
    end_hingle_third[1]-
eps_for_rack_to_rack<tochki_perecechenia_circle_rack_to_rack[1]
&&

tochki_perecechenia_circle_rack_to_rack[1]<end_hingle_third[1]+e
ps_for_rack_to_rack) {
    line(100, 300,
end_hingle_third[0],end_hingle_third[1]);
    fill(0,102,153);
    text("B"+str(int(c)),end_hingle_third[0],
end_hingle_third[1]+20);
    fill(255,255,255);
    }
    }
    c++;

}

    fill(0,102,153);
    text("O1",110, 300);
    fill(255,255,255);
}

```

3.1 Побудова плану прискорень

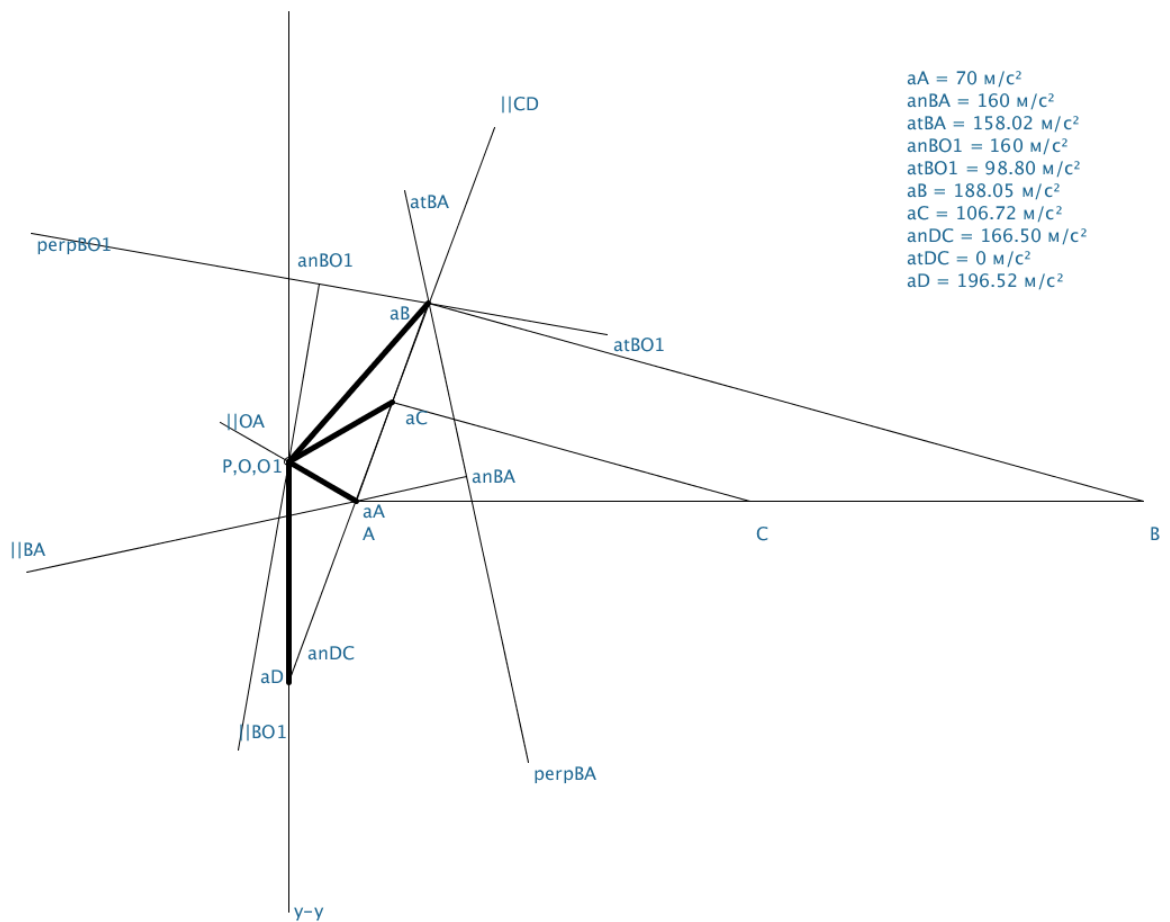


Рисунок. 9 – план прискорень

```

float[] hinge(float x_start, float y_start, float
angle_rotation, float len){
    Знаходять кінцеві координати ланки
    float angle_rotation_rad = radians(angle_rotation);
    float x_end = x_start + len * cos(angle_rotation_rad);
    float y_end = y_start + len * sin(angle_rotation_rad);
    float[] end_coordinates = {x_end, y_end};
    return end_coordinates;
}

void paint_shtrix(float leftX, float leftY, float
leftBetween, float lenShtrix, int count, float angel_rotation,
float povorot){
    float x_start_shtrix, y_start_shtrix;
    float angel_rotation_rad = radians(angel_rotation);
    for( int i=1; i<count; i++){
        x_start_shtrix = leftX + leftBetween * i *
cos(angel_rotation_rad);
        y_start_shtrix = leftY + leftBetween * i *
sin(angel_rotation_rad);
        float[] end_shtrix =
hinge(x_start_shtrix,y_start_shtrix,(angel_rotation+povorot),
lenShtrix);
    }
}

```

```

line(x_start_shtrix,y_start_shtrix,end_shtrix[0],end_shtrix[1]
);
    }
}
void rack(float x_start, float y_start, float
angel_rotation){
    Відмалювання стійки під будь-яким кутом
    int len = 30; // довжина сторони трикутника

    Кінцеві координати трикутника
    float[] right_hinge =
hinge(x_start,y_start,(angel_rotation+60),len);
    float[] left_hinge =
hinge(x_start,y_start,(angel_rotation+120),len);

    Відмалювання трикутника
    line(x_start,y_start,left_hinge[0],left_hinge[1]);
    line(x_start,y_start,right_hinge[0],right_hinge[1]);

line(right_hinge[0],right_hinge[1],left_hinge[0],left_hinge[1]
);
    circle(x_start, y_start, 7);

Нижні рисочки під трикутником
    float len_between_shtrix = len/6.5;
    float len_shtrix = 5;
    int count = 7;
    float povorot = 135;

paint_shtrix(left_hinge[0],left_hinge[1],len_between_shtrix,le
n_shtrix, count,angel_rotation, povorot);
}
void slider(float x_center, float y_center, int
angle_rotation ){
    Відмалювання повзуна під будь-яким кутом
    float len = 30;
прямокутник
    float
right_downX=x_center+len*cos(radians(angle_rotation+30));
    float
right_downY=y_center+len*sin(radians(angle_rotation+30));
    float
left_downX=x_center+len*cos(radians(angle_rotation+150));
    float
left_downY=y_center+len*sin(radians(angle_rotation+150));
    float
left_topX=x_center+len*cos(radians(angle_rotation+210));
    float
left_topY=y_center+len*sin(radians(angle_rotation+210));
    float
right_topX=x_center+len*cos(radians(angle_rotation+330));

```

```

float
right_topY=y_center+len*sin(radians(angle_rotation+330));

line(right_downX,right_downY,left_downX,left_downY); //
довжина ліва
line(left_downX,left_downY,left_topX,left_topY); //
ширина вгору
line(left_topX,left_topY,right_topX,right_topY); //
довжина права
line(right_topX,right_topY,right_downX,right_downY); //
ширина нижня
circle(x_center,y_center,10);

Смужки поруч з прямокутником
float len_2 = 35;
float
right_downX_2=x_center+len_2*cos(radians(angle_rotation+30));
float
right_downY_2=y_center+len_2*sin(radians(angle_rotation+30));
float
left_downX_2=x_center+len_2*cos(radians(angle_rotation+150));
float
left_downY_2=y_center+len_2*sin(radians(angle_rotation+150));
float
left_topX_2=x_center+len_2*cos(radians(angle_rotation+210));
float
left_topY_2=y_center+len_2*sin(radians(angle_rotation+210));
float
right_topX_2=x_center+len_2*cos(radians(angle_rotation+330));
float
right_topY_2=y_center+35*sin(radians(angle_rotation+330));

line(right_downX_2,right_downY_2,left_downX_2,left_downY_2);

line(left_topX_2,left_topY_2,right_topX_2,right_topY_2);

Штрихи
float len_between_shtrix = len_2/6.5;
float len_shtrix = 5;
int count = 11;
int povorot = 60;
int povorot_1 = -120;

paint_shtrix(left_downX_2,left_downY_2,len_between_shtrix,len_
shtrix, count,angle_rotation, povorot);

paint_shtrix(left_topX_2,left_topY_2,len_between_shtrix,len_sh
trix, count,angle_rotation, povorot_1);
}

```

```

float[] tochki_perecechenia_circle(float x1, float y1,
float r1, float x2, float y2, float r2 ){

float d = sqrt( pow(x2-x1,2) + pow(y2-y1,2) );//
Відстань між двома центрами окружности
float a =(pow(r1,2)-pow(r2,2)+pow(d,2))/(2*d);//
відрізок до висоти
float h = sqrt(pow(r1,2)-pow(a,2));// висота
float xh = x1+a*(x2-x1)/d;// координати початку
висоти
float yh = y1+a*(y2-y1)/d;//

float x3 = xh+h*(y2-y1)/d;// координати точки
перетину
float y3 = yh-h*(x2-x1)/d;//

float x4 = xh-h*(y2-y1)/d;//
float y4 = yh+h*(x2-x1)/d;//
float [] res = {x3,y3,x4,y4};
return res;
}
float[] seredini_otrezka(float x1, float y1, float x2,
float y2){
float x = (x1+x2)/2;
float y = (y1+y2)/2;
float[] res = {x,y};
return res;
}
float gradus_otnositelno_glovnou_pramou(float
x_start,float x_end, float len){
Знаходить кут повороту відносно головної осі
float l = degrees(acos((x_start - x_end)/len));
return l;
}

Складаємо формули двох прямих
float A1 = koordinati_nachala_y_otr_1 -
koordinati_consa_y_otr_1;
float B1 = koordinati_consa_x_otr_1 -
koordinati_nachala_x_otr_1;
float C1 =
koordinati_nachala_x_otr_1*koordinati_consa_y_otr_1 -
koordinati_consa_x_otr_1*koordinati_nachala_y_otr_1;
float A2 = koordinati_nachala_y_otr_2 -
koordinati_consa_y_otr_2;
float B2 =koordinati_consa_x_otr_2 -
koordinati_nachala_x_otr_2;
float C2 =
koordinati_nachala_x_otr_2*koordinati_consa_y_otr_2 -
koordinati_consa_x_otr_2*koordinati_nachala_y_otr_2;
Вирішуємо систему двох рівнянь

```



```
float y = (C2*A1 - C1*A2) / (B1*A2 - B2*A1);
float x = (-C1 - B1*y) / A1;
```

перевіряємо, чи знаходиться рішення системи (точка перетину) на першому відрізку, min/max - тому координати точки можуть бути задані не по порядку зростання

```
float[] res = {x,y};
return res;
```

```
}
```

```
void setup(){
  size(1200,900);
  background(255,255,255);
```

```
float x_start = 800;
float y_start =440;
```

Координати похибки

```
float eps_for_rack_to_rack = 0.67;
float eps_for_rack_to_slider = 0.38;
float c = 0;
int angle = 210;
float[] Polus = {300, 450};
```

```
circle(Polus[0],Polus[1], 7);
textSize(16);
fill(0,102,153);
text("P,O,O1",Polus[0]-59,Polus[1]+10 );
fill(255,255,255);
```

Координати кінця ланки навколо головного рядка

```
float[] end_hingle_first =
hinge(x_start,y_start,angle,70);
float[] serVaVb = {};
```

Відмалювання прискорення A

```
float alpha_A =
gradus_otnositelno_glovnou_pramou(x_start,end_hingle_first[0],70
);
```

```
float[] end_hingle_first_s =
hinge(Polus[0],Polus[1],alpha_A,70);
float[] end_hingle_first_s_2 =
hinge(Polus[0],Polus[1],alpha_A+180,70);
strokeWeight(5);
line(Polus[0],Polus[1],
end_hingle_first_s[0],end_hingle_first_s[1]);
strokeWeight(1);
```

```
line(Polus[0],Polus[1],
end_hingle_first_s_2[0],end_hingle_first_s_2[1]);
```

```

fill(0,102,153);

text("aA = 70 м/с2",850 ,115);

text("anBA = 160 м/с2",850 ,135);
text("atBA = 158.02 м/с2",850 ,155);
text("anBO1 = 160 м/с2",850 ,175);
text("atBO1 = 98.80 м/с2",850 ,195);
text("aB = 188.05 м/с2",850 ,215);

text("aC = 106.72 м/с2",850 ,235);

text("anDC = 166.50 м/с2",850 ,255);
text("atDC = 0 м/с2",850 ,275);
text("aD = 196.52 м/с2",850 ,295);

text("aA",end_hingle_first_s[0]+5,end_hingle_first_s[1]+15 );
text("A",end_hingle_first_s[0]+5,end_hingle_first_s[1]+35 );

text("||OA",end_hingle_first_s_2[0]+5,end_hingle_first_s_2[1]+5
);
fill(255,255,255);

```

Знаходить координати перетину кіл двох стійок

```

float[] tochki_perecechenia_circle_rack_to_rack =
tochki_perecechenia_circle(end_hingle_first[0],end_hingle_first[
1],700,100,300,260);// стійка + стійка

```

Знаходить координати середини відрізка

```

float[] koordinati_seredini =
seredini_otrezka(end_hingle_first[0],end_hingle_first[1],
tochki_perecechenia_circle_rack_to_rack[0],tochki_perecechenia_c
ircle_rack_to_rack[1]); // Координати середини для повзуна

```

Цикл зміни кута

```

for(float i=0;i<360;i+=0.3){
float[] end_hinge_second =
hinge(koordinati_seredini[0],koordinati_seredini[1],i,150);
float[] end_hingle_third = hinge(100,300,i,260); //

```

Умовний рух повзуна

Умовний перетин двох кіл з похибкою

```

if( end_hingle_third[0]-
eps_for_rack_to_rack<tochki_perecechenia_circle_rack_to_rack[0]
&&
tochki_perecechenia_circle_rack_to_rack[0]<end_hingle_third[
0]+eps_for_rack_to_rack && end_hingle_third[1]-
eps_for_rack_to_rack<tochki_perecechenia_circle_rack_to_rack[1]

```

```

&&
tochki_perecechenia_circle_rack_to_rack[1]<end_hingle_third[1]+e
ps_for_rack_to_rack) {
    Відмалювання швидкості B01
    float alpha_B =
gradus_otnositelno_glovnou_pramou(end_hingle_third[0],100,260);
    float[] end_hingle_second_s =
hinge(Polus[0],Polus[1],alpha_B,260);

    line(Polus[0],Polus[1],
end_hingle_second_s[0],end_hingle_second_s[1]);

    fill(0,102,153);

text("||B01",end_hingle_second_s[0],end_hingle_second_s[1]-10 );
    fill(255,255,255);

    float[] end_hingle_second_s_v2 =
hinge(Polus[0],Polus[1],alpha_B-180,160);
    float[] end_hingle_second_s_perp_left =
hinge(end_hingle_second_s_v2[0],end_hingle_second_s_v2[1],alpha_
B+90,260);
    float[] end_hingle_second_s_perp_right =
hinge(end_hingle_second_s_v2[0],end_hingle_second_s_v2[1],alpha_
B-90,260);

line(end_hingle_second_s_v2[0],end_hingle_second_s_v2[1],
end_hingle_second_s_perp_left[0],end_hingle_second_s_perp_left[1
]);

line(end_hingle_second_s_v2[0],end_hingle_second_s_v2[1],
end_hingle_second_s_perp_right[0],end_hingle_second_s_perp_right
[1]);

    line(Polus[0],Polus[1],
end_hingle_second_s_v2[0],end_hingle_second_s_v2[1]);
    textSize(16);
    fill(0,102,153);
    text("anB01",end_hingle_second_s_v2[0]-
20,end_hingle_second_s_v2[1]-15 );
    fill(255,255,255);

    Відмалювання швидкості BA
    float alpha_B_2 =
gradus_otnositelno_glovnou_pramou(end_hingle_third[0],end_hingle
_first[0],700);
    float[] end_hingle_second_s_2 =
hinge(end_hingle_first_s[0],end_hingle_first_s[1],alpha_B_2,300)
;

line(end_hingle_first_s[0],end_hingle_first_s[1],
end_hingle_second_s_2[0],end_hingle_second_s_2[1]);
    fill(0,102,153);

```

```

        text("||BA",end_hingle_second_s_2[0]-
15,end_hingle_second_s_2[1]-15 );
        fill(255,255,255);

        float[] end_hingle_second_s_v2_2 =
hinge(end_hingle_first_s[0],end_hingle_first_s[1],alpha_B_2-
180,100);

line(end_hingle_first_s[0],end_hingle_first_s[1],
end_hingle_second_s_v2_2[0],end_hingle_second_s_v2_2[1]);

        float[] end_hingle_second_s_v2_2_perp_left
=
hinge(end_hingle_second_s_v2_2[0],end_hingle_second_s_v2_2[1],al
pha_B_2+90,260);
        float[] end_hingle_second_s_v2_2_perp_right
=
hinge(end_hingle_second_s_v2_2[0],end_hingle_second_s_v2_2[1],al
pha_B_2-90,260);

line(end_hingle_second_s_v2_2[0],end_hingle_second_s_v2_2[1],
end_hingle_second_s_v2_2_perp_left[0],end_hingle_second_s_v2_2_p
erp_left[1]);

line(end_hingle_second_s_v2_2[0],end_hingle_second_s_v2_2[1],
end_hingle_second_s_v2_2_perp_right[0],end_hingle_second_s_v2_2_
perp_right[1]);

        textSize(16);
        fill(0,102,153);
text("anBA",end_hingle_second_s_v2_2[0]+5,end_hingle_second_s_v2
_2[1]+5 );
        fill(255,255,255);

        float[] perecechenieB =
tochki_perecechenia_dvyx_pramix(end_hingle_second_s_v2[0],end_hi
ngle_second_s_v2[1],
end_hingle_second_s_perp_right[0],end_hingle_second_s_perp_right
[1], end_hingle_second_s_v2_2[0],end_hingle_second_s_v2_2[1],
end_hingle_second_s_v2_2_perp_right[0],end_hingle_second_s_v2_2_
perp_right[1] );
        strokeWeight(5);

line(Polus[0],Polus[1],perecechenieB[0],perecechenieB[1]);
        strokeWeight(1);
        fill(0,102,153);
        text("aB",perecechenieB[0]-
35,perecechenieB[1]+15 );

text("atB01",end_hingle_second_s_perp_right[0]+5,end_hingle_seco
nd_s_perp_right[1]+15);

text("perpB01",end_hingle_second_s_perp_left[0]+5,end_hingle_sec
ond_s_perp_left[1]+15);

```

```
        text("atBA",
end_hingle_second_s_v2_2_perp_left[0]+5,end_hingle_second_s_v2_2
_perp_left[1]+15 );
```

```
text("perpBA",end_hingle_second_s_v2_2_perp_right[0]+5,end_hingl
e_second_s_v2_2_perp_right[1]+15);
        fill(255,255,255);
```

Пряма АВ

Відмалювання прямої АВ

```
float[] konetcAB =
hinge(end_hingle_first_s[0],end_hingle_first_s[1],0,700);

line(end_hingle_first_s[0],end_hingle_first_s[1],konetcAB[0],kon
etcAB[1]);
```

```
line(perecechenieB[0],perecechenieB[1],konetcAB[0],konetcAB[1]);
```

```
line(perecechenieB[0],perecechenieB[1],end_hingle_first_s[0],end
_hingle_first_s[1]);
```

```
float[] konetcAC =
hinge(end_hingle_first_s[0],end_hingle_first_s[1],0,350);
```

```
float[] serVAVB =
seredini_otrezka(perecechenieB[0],perecechenieB[1],
end_hingle_first_s[0],end_hingle_first_s[1]);
    serVaVb = serVAVB;
    line(konetcAC[0],konetcAC[1],
serVAVB[0],serVAVB[1]);
    strokeWeight(5);
```

```
line(Polus[0],Polus[1],serVAVB[0],serVAVB[1]);
    strokeWeight(1);
    fill(0,102,153);
    text("aC",serVAVB[0]+11,serVAVB[1]+20 );
    text("C",konetcAC[0]+5,konetcAC[1]+35 );
    text("B",konetcAB[0]+5,konetcAB[1]+35 );
    fill(255,255,255);
    }
```

```
        if(end_hinge_second[0]>450-
eps_for_rack_to_slider &&
end_hinge_second[0]<450+eps_for_rack_to_slider){
```

Відкидаємо нижні координати

```
if(end_hinge_second[1]<koordinati_seredini[1]){
```

```
line(Polus[0],Polus[1],Polus[0],Polus[1]+400);
    line(Polus[0],Polus[1],Polus[0],Polus[1]-
400);

    textSize(16);
    fill(0,102,153);
    text("y-y",Polus[0]+5,Polus[1]+405 );
```

```

        fill(255,255,255);

        float alpha_D =
gradus_otnositelno_glovnou_pramou(koordinati_seredini[0],end_hin
ge_second[0],150);
        float[] end_hingle_third_s_right =
hinge(serVaVb[0],serVaVb[1],alpha_D,260);
        float[] end_hingle_third_s_left =
hinge(serVaVb[0],serVaVb[1],alpha_D-180,260);
        line(serVaVb[0],serVaVb[1],
end_hingle_third_s_right[0],end_hingle_third_s_right[1]);
        line(serVaVb[0],serVaVb[1],
end_hingle_third_s_left[0],end_hingle_third_s_left[1]);
        fill(0,102,153);

text("||CD",end_hingle_third_s_left[0]+5,end_hingle_third_s_left
[1]-15 );

text("anDC",end_hingle_third_s_right[0]+15,end_hingle_third_s_ri
ght[1]-15 );
        fill(255,255,255);

        float[] perecechenieD =
tochki_perecechenia_dvyx_pramix(serVaVb[0],serVaVb[1],
end_hingle_third_s_right[0],end_hingle_third_s_right[1],Polus[0]
,Polus[1], Polus[0],Polus[1]+400 );
        strokeWeight(5);

line(Polus[0],Polus[1],perecechenieD[0],perecechenieD[1]);
        fill(0,102,153);
        text("aD",perecechenieD[0]-
25,perecechenieD[1] );
        fill(255,255,255);

        }
    }
}

```

3.2 Побудова плану швидкостей

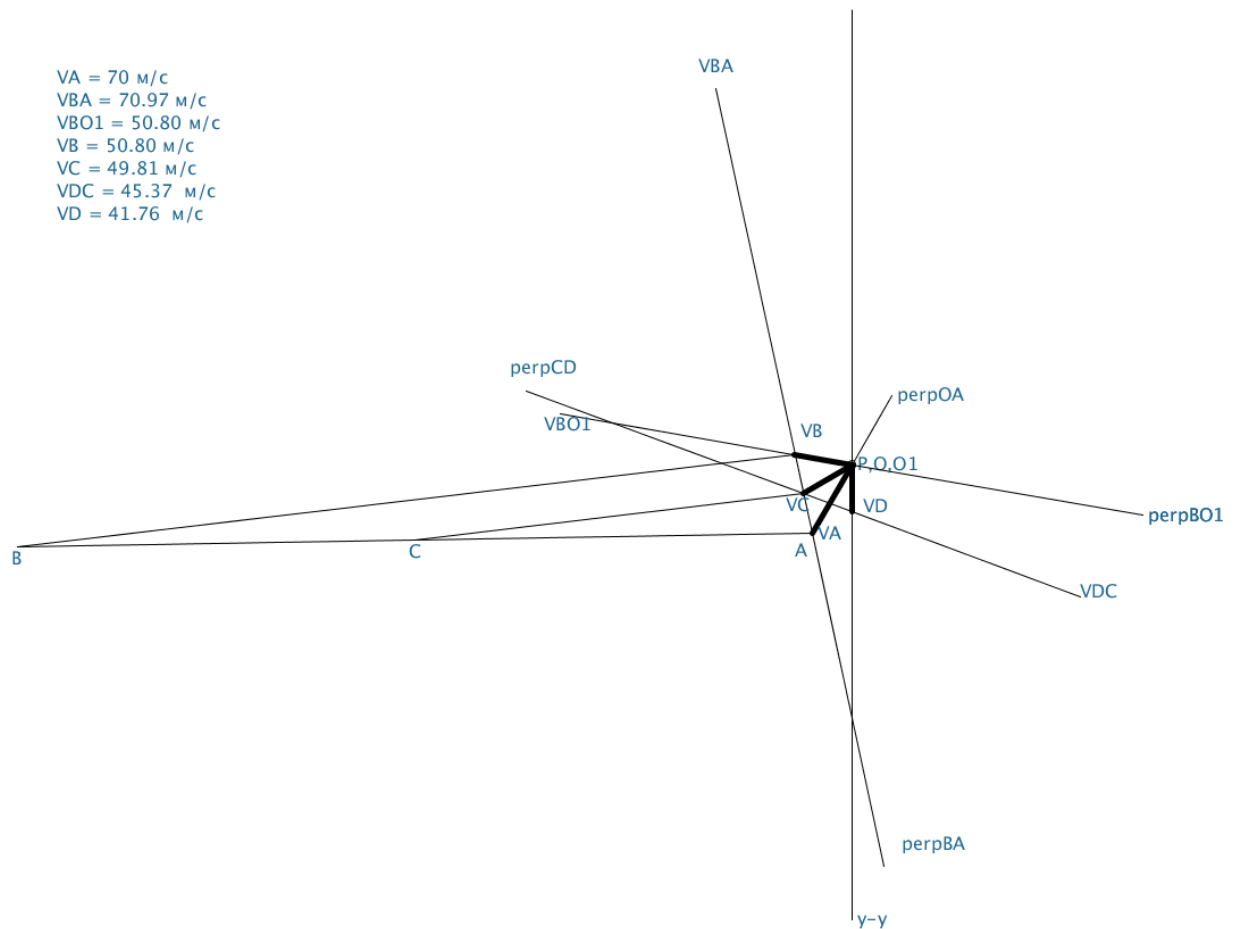


Рисунок 10 – план швидкостей

```

float[] hinge(float x_start, float y_start, float
angle_rotation, float len){
    Знаходить кінцеві координати ланки
    float angle_rotation_rad = radians(angle_rotation);
    float x_end = x_start + len * cos(angle_rotation_rad);
    float y_end = y_start + len * sin(angle_rotation_rad);
    float[] end_coordinates = {x_end, y_end};
    return end_coordinates;
}

void paint_shtrix(float leftX, float leftY, float
leftBetween, float lenShtrix, int count, float angel_rotation,
float povorot){
    float x_start_shtrix, y_start_shtrix;
    float angel_rotation_rad = radians(angel_rotation);
    for( int i=1; i<count; i++){
        x_start_shtrix = leftX + leftBetween * i *
cos(angel_rotation_rad);
        y_start_shtrix = leftY + leftBetween * i *
sin(angel_rotation_rad);
        float[] end_shtrix =
hinge(x_start_shtrix,y_start_shtrix,(angel_rotation+povorot),
lenShtrix);

line(x_start_shtrix,y_start_shtrix,end_shtrix[0],end_shtrix[1]);

```

```

    }
}

void rack(float x_start, float y_start, float
angel_rotation){
    Відмалювання стійки під будь-яким кутом
    int len = 30; // довжина сторони трикутника

    Кінцеві координати трикутника
    float[] right_hinge =
hinge(x_start,y_start,(angel_rotation+60),len);
    float[] left_hinge =
hinge(x_start,y_start,(angel_rotation+120),len);

    Відмалювання трикутника
    line(x_start,y_start,left_hinge[0],left_hinge[1]);
    line(x_start,y_start,right_hinge[0],right_hinge[1]);

line(right_hinge[0],right_hinge[1],left_hinge[0],left_hinge[1]);
    circle(x_start, y_start, 7);

    Нижні рисочки під трикутником
    float len_between_shtrix = len/6.5;
    float len_shtrix = 5;
    int count = 7;
    float povorot = 135;

paint_shtrix(left_hinge[0],left_hinge[1],len_between_shtrix,len_
shtrix, count,angel_rotation, povorot);
}

void slider(float x_center, float y_center, int
angle_rotation ){
    Відмалювання повзуна під будь-яким кутом
    float len = 30;
    Прямокутник
    float
right_downX=x_center+len*cos(radians(angle_rotation+30));
    float
right_downY=y_center+len*sin(radians(angle_rotation+30));
    float
left_downX=x_center+len*cos(radians(angle_rotation+150));
    float
left_downY=y_center+len*sin(radians(angle_rotation+150));
    float
left_topX=x_center+len*cos(radians(angle_rotation+210));
    float
left_topY=y_center+len*sin(radians(angle_rotation+210));
    float
right_topX=x_center+len*cos(radians(angle_rotation+330));
    float
right_topY=y_center+len*sin(radians(angle_rotation+330));

```



```

        line(right_downX,right_downY,left_downX,left_downY); //
довжина ліва
        line(left_downX,left_downY,left_topX,left_topY); //
ширина вгору
        line(left_topX,left_topY,right_topX,right_topY);//
довжина права
        line(right_topX,right_topY,right_downX,right_downY);//
ширина нижня
        circle(x_center,y_center,10);
Смужки поруч з прямокутником
        float len_2 = 35;
        float
right_downX_2=x_center+len_2*cos(radians(angle_rotation+30));
        float
right_downY_2=y_center+len_2*sin(radians(angle_rotation+30));
        float
left_downX_2=x_center+len_2*cos(radians(angle_rotation+150));
        float
left_downY_2=y_center+len_2*sin(radians(angle_rotation+150));
        float
left_topX_2=x_center+len_2*cos(radians(angle_rotation+210));
        float
left_topY_2=y_center+len_2*sin(radians(angle_rotation+210));
        float
right_topX_2=x_center+len_2*cos(radians(angle_rotation+330));
        float
right_topY_2=y_center+35*sin(radians(angle_rotation+330));

line(right_downX_2,right_downY_2,left_downX_2,left_downY_2);
        line(left_topX_2,left_topY_2,right_topX_2,right_topY_2);

Штрихи
        float len_between_shtrix = len_2/6.5;
        float len_shtrix = 5;
        int count = 11;
        int povorot = 60;
        int povorot_1 = -120;

paint_shtrix(left_downX_2,left_downY_2,len_between_shtrix,len_shtrix,
count,angle_rotation, povorot);

paint_shtrix(left_topX_2,left_topY_2,len_between_shtrix,len_shtrix,
count,angle_rotation, povorot_1);
    }

        float[] tochki_perecechenia_circle(float x1, float y1, float
r1, float x2, float y2, float r2 ){

        float d = sqrt( pow(x2-x1,2) + pow(y2-y1,2) );//
Відстань між двома центрами окружності

```

```

float a =(pow(r1,2)-pow(r2,2)+pow(d,2))/(2*d);//
Відрізок до висоти
float h = sqrt(pow(r1,2)-pow(a,2));// висота
float xh = x1+a*(x2-x1)/d;// координати початку
висоти
float yh = y1+a*(y2-y1)/d;

float x3 = xh+h*(y2-y1)/d;// координати точки
перетину
float y3 = yh-h*(x2-x1)/d;

float x4 = xh-h*(y2-y1)/d;
float y4 = yh+h*(x2-x1)/d;
float [] res = {x3,y3,x4,y4};
return res;
}

float[] seredini_otrezka(float x1, float y1, float x2, float
y2){
float x = (x1+x2)/2;
float y = (y1+y2)/2;
float[] res = {x,y};
return res;
}
float gradus_otnositelno_glovnou_pramou(float x_start,float
x_end, float len){
Знаходить кут повороту відносно головної осі
float l = degrees(acos((x_start - x_end)/len));
return l;
}

float[] tochki_perecechenia_dvux_pramix(float
koordinati_nachala_x_otr_1, float koordinati_nachala_y_otr_1,
float
koordinati_consa_x_otr_1, float koordinati_consa_y_otr_1,
float
koordinati_nachala_x_otr_2, float koordinati_nachala_y_otr_2,
float
koordinati_consa_x_otr_2, float koordinati_consa_y_otr_2){

Складаємо формули двох прямих
float A1 = koordinati_nachala_y_otr_1 -
koordinati_consa_y_otr_1;
float B1 = koordinati_consa_x_otr_1 -
koordinati_nachala_x_otr_1;
float C1 =
koordinati_nachala_x_otr_1*koordinati_consa_y_otr_1 -
koordinati_consa_x_otr_1*koordinati_nachala_y_otr_1;
float A2 = koordinati_nachala_y_otr_2 -
koordinati_consa_y_otr_2;
float B2 =koordinati_consa_x_otr_2 -
koordinati_nachala_x_otr_2;

```

```

float C2 =
koordinati_nachala_x_otr_2*koordinati_consa_y_otr_2 -
koordinati_consa_x_otr_2*koordinati_nachala_y_otr_2;

```

Вирішуємо систему двох рівнянь

```

float y = (C2*A1 - C1*A2) / (B1*A2 - B2*A1);
float x = (-C1 - B1*y) / A1;

```

перевіряємо, чи знаходиться рішення системи (точка перетину) на першому відрізку, min/max – тому що координати точки можуть бути задані не по порядку зростання

```

float[] res = {x,y};
return res;

```

```

}

```

```

void setup() {
  size(1200,900);
  background(255,255,255);

  float x_start = 800;
  float y_start =440;
  float[] Polus = {800, 450};

```

```

  circle(Polus[0],Polus[1], 7);
  textSize(16);
  fill(0,102,153);
  text("P,O,O1",Polus[0]+5,Polus[1]+5 );
  fill(255,255,255);

```

Координати похибки

```

float eps_for_rack_to_rack = 0.67;
float eps_for_rack_to_slider = 0.38;

```

```

int angle = 210;

```

Координати кінця ланки навколо головної стійки

```

float[] end_hingle_first =
hinge(x_start,y_start,angle,70);

```

Відмалювання швидкості A

```

float alpha_A =
gradus_otnositelno_glovnou_pramou(x_start,end_hingle_first[0],70
);

```

```

float[] end_hingle_first_s =
hinge(Polus[0],Polus[1],alpha_A+90,70);

```

```

strokeWeight(5);
line(Polus[0],Polus[1],
end_hingle_first_s[0],end_hingle_first_s[1]);
strokeWeight(1);

```

```

fill(0,102,153);

text("VA",end_hingle_first_s[0]+5,end_hingle_first_s[1]+5 );
    text("A",end_hingle_first_s[0]-
15,end_hingle_first_s[1]+20 );

    text("VA = 70 м/с",100 ,115);
    text("VBA = 70.97 м/с  ",100 ,135);
    text("VBO1 = 50.80 м/с",100 ,155);
    text("VB = 50.80 м/с",100 ,175);
    text("VC = 49.81 м/с",100 ,195);
    text("VDC = 45.37 м/с",100 ,215);
    text("VD = 41.76 м/с",100 ,235);

    fill(255,255,255);
    float[] end_hingle_first_s_v2 =
hinge(Polus[0],Polus[1],alpha_A-90,70);
    line(Polus[0],Polus[1],
end_hingle_first_s_v2[0],end_hingle_first_s_v2[1]);
    textSize(16);
    fill(0,102,153);

text("perpOA",end_hingle_first_s_v2[0]+5,end_hingle_first_s_v2[1]
+5 );
    fill(255,255,255);

float[] serVaVb = {};

```

Знаходить координати перетину кіл двох стійок

```

float[] tocki_perecechenia_circle_rack_to_rack =
tocki_perecechenia_circle(end_hingle_first[0],end_hingle_first[
1],700,100,300,260);// стійка + стійка

```

Знаходить координати середини відрізка

```

float[] koordinati_seredini =
seredini_otrezka(end_hingle_first[0],end_hingle_first[1],
tocki_perecechenia_circle_rack_to_rack[0],tocki_perecechenia_c
ircle_rack_to_rack[1]); // координати середини для повзуна

```

Цикл зміни кута

```

for(float i=0;i<360;i+=0.3){
    float[] end_hinge_second =
hinge(koordinati_seredini[0],koordinati_seredini[1],i,150);
    float[] end_hingle_third = hinge(100,300,i,260);

```

Умова перетин двох кіл з похибкою

```

if( end_hingle_third[0]-
eps_for_rack_to_rack<tocki_perecechenia_circle_rack_to_rack[0]
&&
    tocki_perecechenia_circle_rack_to_rack[0]<end_hingle_third[
0]+eps_for_rack_to_rack && end_hingle_third[1]-

```

```

eps_for_rack_to_rack<tochki_perecechenia_circle_rack_to_rack[1]
&&
    tochki_perecechenia_circle_rack_to_rack[1]<end_hingle_third[
1]+eps_for_rack_to_rack)  {

```

Відмалювання швидкості В01

```

    float alpha_B =
gradus_otnositelno_glovnou_pramou(end_hingle_third[0],100,260);
    float[] end_hingle_second_s =
hinge(Polus[0],Polus[1],alpha_B+90,260);
    line(Polus[0],Polus[1],
end_hingle_second_s[0],end_hingle_second_s[1]);

    fill(0,102,153);
    text("VBO1",end_hingle_second_s[0]-
15,end_hingle_second_s[1]+15 );
    fill(255,255,255);

    float[] end_hingle_second_s_v2 =
hinge(Polus[0],Polus[1],alpha_B-90,260);
    line(Polus[0],Polus[1],
end_hingle_second_s_v2[0],end_hingle_second_s_v2[1]);
    textSize(16);
    fill(0,102,153);

text("perpBO1",end_hingle_second_s_v2[0]+5,end_hingle_second_s_v
2[1]+5 );
    fill(255,255,255);

```

Відмалювання швидкості ВА

```

    float alpha_B_2 =
gradus_otnositelno_glovnou_pramou(end_hingle_third[0],end_hingle
_first[0],700);
    float[] end_hingle_second_s_2 =
hinge(end_hingle_first_s[0],end_hingle_first_s[1],alpha_B_2+90,4
00);

line(end_hingle_first_s[0],end_hingle_first_s[1],
end_hingle_second_s_2[0],end_hingle_second_s_2[1]);
    fill(0,102,153);
    text("VBA",end_hingle_second_s_2[0]-
15,end_hingle_second_s_2[1]-15 );

    fill(255,255,255);

    float[] end_hingle_second_s_v2_2 =
hinge(end_hingle_first_s[0],end_hingle_first_s[1],alpha_B_2-
90,300);

line(end_hingle_first_s[0],end_hingle_first_s[1],
end_hingle_second_s_v2_2[0],end_hingle_second_s_v2_2[1]);

    fill(0,102,153);

```

```

text("perpBA",end_hingle_second_s_v2_2[0]+15,end_hingle_second_s_v2_2[1]-15 );

fill(255,255,255);
textSize(16);
fill(0,102,153);

text("perpBO1",end_hingle_second_s_v2[0]+5,end_hingle_second_s_v2[1]+5 );

fill(255,255,255);

float[] perecechenieB =
tochki_perecechenia_dvyx_pramix(Polus[0],Polus[1],end_hingle_second_s[0],end_hingle_second_s[1],end_hingle_first_s[0],end_hingle_first_s[1],end_hingle_second_s_2[0],end_hingle_second_s_2[1]);

strokeWeight(5);

line(Polus[0],Polus[1],perecechenieB[0],perecechenieB[1]);
strokeWeight(1);
fill(0,102,153);

text("VB",perecechenieB[0]+5,perecechenieB[1]-15 );
fill(255,255,255);

Відмалювання прямої АВ
float[] konetcAB =
hinge(end_hingle_first_s[0],end_hingle_first_s[1],-181,700);

line(end_hingle_first_s[0],end_hingle_first_s[1],konetcAB[0],konetcAB[1]);

line(perecechenieB[0],perecechenieB[1],konetcAB[0],konetcAB[1]);
float[] konetcAC =
hinge(end_hingle_first_s[0],end_hingle_first_s[1],-181,350);
textSize(16);
fill(0,102,153);
text("C",konetcAC[0]-5,konetcAC[1]+15 );
text("B",konetcAB[0]-5,konetcAB[1]+15 );
fill(255,255,255);
float[] serVAVB =
seredini_otrezka(perecechenieB[0],perecechenieB[1],end_hingle_first_s[0],end_hingle_first_s[1]);
serVaVb = serVAVB;
line(konetcAC[0],konetcAC[1],serVAVB[0],serVAVB[1]);
strokeWeight(5);

line(Polus[0],Polus[1],serVAVB[0],serVAVB[1]);
strokeWeight(1);
fill(0,102,153);
text("VC",serVAVB[0]-15,serVAVB[1]+15 );

```

```

        fill(255,255,255);
    }
    Умова руху повзуна
        if(end_hinge_second[0]>450-
eps_for_rack_to_slider &&
end_hinge_second[0]<450+eps_for_rack_to_slider){
        Відкидаємо нижні координати
        if(end_hinge_second[1]<koordinati_seredini[1]){

            line(Polus[0],Polus[1],Polus[0],Polus[1]+400);
            line(Polus[0],Polus[1],Polus[0],Polus[1]-400);
            textSize(16);
            fill(0,102,153);
            text("y-y",Polus[0]+5,Polus[1]+405 );
            fill(255,255,255);
        Відмалювання швидкості CD
            float alpha_D =
gradus_otnositelno_glovnou_pramou(koordinati_seredini[0],end_hin
ge_second[0],150);
                float[] end_hingle_third_s =
hinge(serVaVb[0],serVaVb[1],alpha_D+90,260);
                line(serVaVb[0],serVaVb[1],
end_hingle_third_s[0],end_hingle_third_s[1]);
                fill(0,102,153);
                text("perpCD",end_hingle_third_s[0]-
15,end_hingle_third_s[1]-15 );
                fill(255,255,255);
                float[] end_hingle_third_s_v2 =
hinge(serVaVb[0],serVaVb[1],alpha_D-90,260);
                line(serVaVb[0],serVaVb[1],
end_hingle_third_s_v2[0],end_hingle_third_s_v2[1]);
                float[] perecechenieD =
tochki_perecechenia_dvyx_pramix(serVaVb[0],serVaVb[1],
end_hingle_third_s_v2[0],end_hingle_third_s_v2[1],Polus[0],Polus
[1], Polus[0],Polus[1]+400 );
                strokeWeight(5);

            line(Polus[0],Polus[1],perecechenieD[0],perecechenieD[1]);
                fill(0,102,153);

            text("VDC",end_hingle_third_s_v2[0],end_hingle_third_s_v2[1] );

            text("VD",perecechenieD[0]+10,perecechenieD[1] );
                fill(255,255,255);

            print(dist(perecechenieD[0],perecechenieD[1],Polus[0],Polus[1]))
;
        }
    }
}

```

3.3 Анімація механізму

Натисніть на клавіатурі кнопку Up і анімація завершиться при повторному натисканні анімація почнеться заново

Натисніть на клавіатурі кнопку Down і анімація завершиться при повторному натисканні анімація продовже свій рух

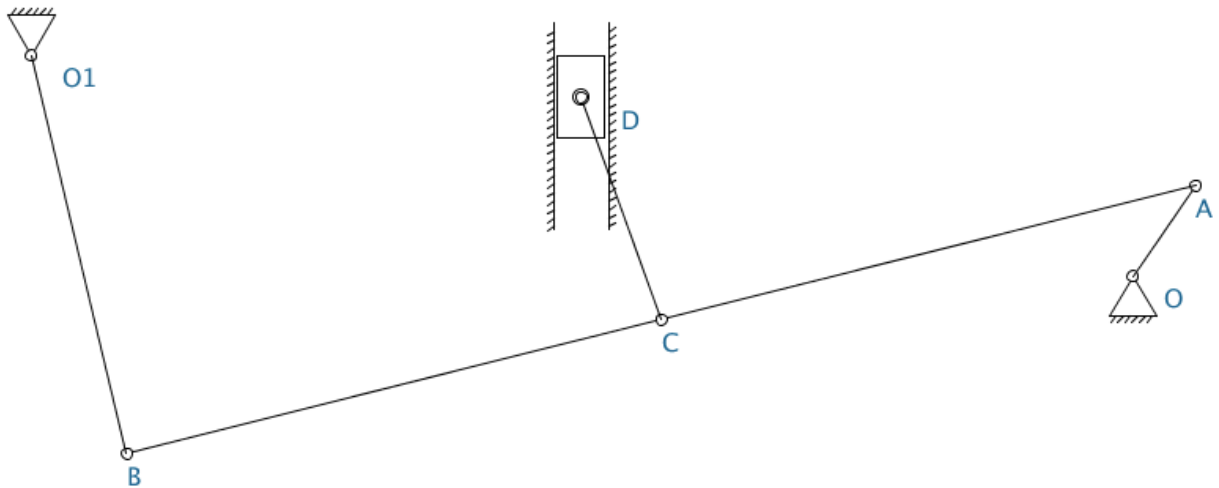


Рисунок 11 – анімація механізму

```
boolean check = true;
void keyPressed() {
  if (key == CODED) {
    if (keyCode == UP) {
      if (check){
        angle=0;
        redraw();
        noLoop();
        check=false;
      }else{
        loop();
        check=true;
      }
    }
    else if (keyCode == DOWN) {
      if (check){

        noLoop();
        check=false;
      }else{
        loop();
        check=true;
      }
    }
  }
}
```

```
float[] hinge(float x_start, float y_start, float
angle_rotation, float len){
```



```

Знаходить кінцеві координати ланки
float angle_rotation_rad = radians(angle_rotation);
float x_end = x_start + len * cos(angle_rotation_rad);
float y_end = y_start + len * sin(angle_rotation_rad);
float[] end_coordinates = {x_end, y_end};
return end_coordinates;
}

void paint_shtrix(float leftX, float leftY, float
leftBetween, float lenShtrix, int count, float angel_rotation,
float povorot){
    float x_start_shtrix, y_start_shtrix;
    float angel_rotation_rad = radians(angel_rotation);
    for( int i=1; i<count; i++){
        x_start_shtrix = leftX + leftBetween * i *
cos(angel_rotation_rad);
        y_start_shtrix = leftY + leftBetween * i *
sin(angel_rotation_rad);
        float[] end_shtrix =
hinge(x_start_shtrix,y_start_shtrix,(angel_rotation+povorot),
lenShtrix);

line(x_start_shtrix,y_start_shtrix,end_shtrix[0],end_shtrix[1]);
    }
}

void rack(float x_start, float y_start, float
angel_rotation){
    Відмалювання стійки під будь-яким кутом
    int len = 30; // довжина сторони трикутника

    Кінцеві координати трикутника
    float[] right_hinge =
hinge(x_start,y_start,(angel_rotation+60),len);
    float[] left_hinge =
hinge(x_start,y_start,(angel_rotation+120),len);

    Відмалювання трикутника
    line(x_start,y_start,left_hinge[0],left_hinge[1]);
    line(x_start,y_start,right_hinge[0],right_hinge[1]);

line(right_hinge[0],right_hinge[1],left_hinge[0],left_hinge[1]);
    circle(x_start, y_start, 7);

    Нижні рисочки під трикутником
    float len_between_shtrix = len/6.5;
    float len_shtrix = 5;
    int count = 7;
    float povorot = 135;

paint_shtrix(left_hinge[0],left_hinge[1],len_between_shtrix,len_
shtrix, count,angel_rotation, povorot);
}

```

```

void slider(float x_center, float y_center, int
angle_rotation ){
    Відмалювання повзун під будь-яким кутом
    float len = 30;
    Прямокутник
    float
right_downX=x_center+len*cos(radians(angle_rotation+30));
    float
right_downY=y_center+len*sin(radians(angle_rotation+30));
    float
left_downX=x_center+len*cos(radians(angle_rotation+150));
    float
left_downY=y_center+len*sin(radians(angle_rotation+150));
    float
left_topX=x_center+len*cos(radians(angle_rotation+210));
    float
left_topY=y_center+len*sin(radians(angle_rotation+210));
    float
right_topX=x_center+len*cos(radians(angle_rotation+330));
    float
right_topY=y_center+len*sin(radians(angle_rotation+330));

    line(right_downX,right_downY,left_downX,left_downY); //
довжина ліва
    line(left_downX,left_downY,left_topX,left_topY); //
ширина верхня
    line(left_topX,left_topY,right_topX,right_topY);//
довжина права
    line(right_topX,right_topY,right_downX,right_downY);//
ширина нижня
    circle(x_center,y_center,10);
}

void stoiki(float x_center, float y_center, int
angle_rotation){
    Смужки поруч з прямокутником
    float len_2 = 35;
    float
right_downX_2=x_center+len_2*cos(radians(angle_rotation+30));
    float
right_downY_2=y_center+len_2*sin(radians(angle_rotation+30));
    float
left_downX_2=x_center+len_2*cos(radians(angle_rotation+150));
    float
left_downY_2=y_center+len_2*sin(radians(angle_rotation+150));
    float
left_topX_2=x_center+len_2*cos(radians(angle_rotation+210));
    float
left_topY_2=y_center+len_2*sin(radians(angle_rotation+210));
}

```

```

float
right_topX_2=x_center+len_2*cos(radians(angle_rotation+330));
float
right_topY_2=y_center+35*sin(radians(angle_rotation+330));
line(right_downX_2,right_downY_2+70,left_downX_2,left_downY_2);
line(left_topX_2,left_topY_2,right_topX_2,right_topY_2+70);

Штрихи
float len_between_shtrix = len_2/6.5;
float len_shtrix = 5;
int count = 25;
int povorot = 60;
int povorot_1 = -120;

paint_shtrix(left_downX_2,left_downY_2,len_between_shtrix,len_shtrix, count,angle_rotation, povorot);

paint_shtrix(left_topX_2,left_topY_2,len_between_shtrix,len_shtrix, count,angle_rotation, povorot_1);
}

float[] tochki_perecechenia_circle(float x1, float y1, float r1, float x2, float y2, float r2 ){

float d = sqrt( pow(x2-x1,2) + pow(y2-y1,2) );//
Відстань між двома центрами окружності
float a =(pow(r1,2)-pow(r2,2)+pow(d,2))/(2*d);//
відрізок до висоти
float h = sqrt(pow(r1,2)-pow(a,2));// висота
float xh = x1+a*(x2-x1)/d;//координати початку
висоти
float yh = y1+a*(y2-y1)/d;//

float x3 = xh+h*(y2-y1)/d;//координати точки перетину
float y3 = yh-h*(x2-x1)/d;//

float x4 = xh-h*(y2-y1)/d;//
float y4 = yh+h*(x2-x1)/d;//
float [] res = {x3,y3,x4,y4};
return res;
}

float[] seredini_otrezka(float x1, float y1, float x2, float y2){
float x = (x1+x2)/2;
float y = (y1+y2)/2;
float[] res = {x,y};
return res;
}

```

```

int angle = 0;
void draw(){

    background(255,255,255);
    stoiki(449.86868, 310,90);
    textSize(16);
Координати головної стійки
    float x_start = 800;
    float y_start =440;

    rack(x_start,y_start,0);// Головна стійка
    rack(100,300,180);// Друга стійка
        fill(0,102,153);
        text("01",120,300+20 );
        text("0",x_start+20,y_start+20 );
        fill(255,255,255);

Координати похибки
    float eps_for_rack_to_rack = 0.67;
    float eps_for_rack_to_slider = 0.38;
Координати кінця ланки навколо головної стійки
    float[] end_hingle_first =
hinge(x_start,y_start,angle,70);
    line(x_start, y_start,
end_hingle_first[0],end_hingle_first[1]);// Відмалювання
ланки навколо головної стійки
    circle(end_hingle_first[0],end_hingle_first[1], 7);
        fill(0,102,153);

    text("A",end_hingle_first[0],end_hingle_first[1]+20 );
        fill(255,255,255);

Знаходить координати перетину кіл двох стійок
    float[] toчки_perecechenia_circle_rack_to_rack =
toчки_perecechenia_circle(end_hingle_first[0],end_hingle_first[
1],700,100,300,260);// стійка + стійка
Відмалювання ліній
    line(end_hingle_first[0],end_hingle_first[1],
toчки_perecechenia_circle_rack_to_rack[0],toчки_perecechenia_c
ircle_rack_to_rack[1]);

    circle(toчки_perecechenia_circle_rack_to_rack[0],toчки_perecec
henia_circle_rack_to_rack[1],7);

Знаходить координати середини відрізка
    float[] koordinati_seredini =
seredini_otrezka(end_hingle_first[0],end_hingle_first[1],
toчки_perecechenia_circle_rack_to_rack[0],toчки_perecechenia_c
ircle_rack_to_rack[1]); // Координати середини для повзуна
Відмальовує по середині окружності
    circle(koordinati_seredini[0],koordinati_seredini[1],7);

Цикл зміни кута

```

```

    for(float i=0;i<360;i+=0.3){
        float[] end_hinge_second =
hinge(koordinati_seredini[0],koordinati_seredini[1],i,150);
        float[] end_hingle_third = hinge(100,300,i,260);
            Умова руху повзуна
            if(end_hinge_second[0]>450-eps_for_rack_to_slider
&& end_hinge_second[0]<450+eps_for_rack_to_slider){
                Відкидаємо нижні координати
                if(end_hinge_second[1]<koordinati_seredini[1]){

slider(end_hinge_second[0],end_hinge_second[1],90);// Повзун

line(koordinati_seredini[0],koordinati_seredini[1],
end_hinge_second[0],end_hinge_second[1]);

                circle(end_hinge_second[0],end_hinge_second[1],7);

                    fill(0,102,153);

text("C",koordinati_seredini[0],koordinati_seredini[1]+20 );

text("D",end_hinge_second[0]+25,end_hinge_second[1]+20 );
                    fill(255,255,255);
                }

            }
            // Умова перетину двох кіл з похибкою
            if( end_hingle_third[0]-
eps_for_rack_to_rack<tochki_perecechenia_circle_rack_to_rack[0]
&&
                tochki_perecechenia_circle_rack_to_rack[0]<end_hingle_third[
0]+eps_for_rack_to_rack && end_hingle_third[1]-
eps_for_rack_to_rack<tochki_perecechenia_circle_rack_to_rack[1]
&&
                tochki_perecechenia_circle_rack_to_rack[1]<end_hingle_third[
1]+eps_for_rack_to_rack) {

                    line(100, 300,
end_hingle_third[0],end_hingle_third[1]);
                    fill(0,102,153);

text("B",end_hingle_third[0],end_hingle_third[1]+20 );
                    fill(255,255,255);
                }

            }
            angle --;
            fill(0,102,153);
            text("Натисніть на клавіатурі кнопку Up і анімація
завершиться ", 120, 100);
            text("при повторному натисканні анімація почнеться
заново", 120, 120);

```

```
        text("Натисніть на клавіатурі кнопку Dow і анімація  
завершиться ", 120, 150);  
        text("при повторному натисканні анімація продовже свій  
рух", 120, 170);  
        fill(255,255,255);  
    }  
  
void setup(){  
    size(900,900);  
  
}
```

4 Побудова 12 положень механізму в SolidWorks

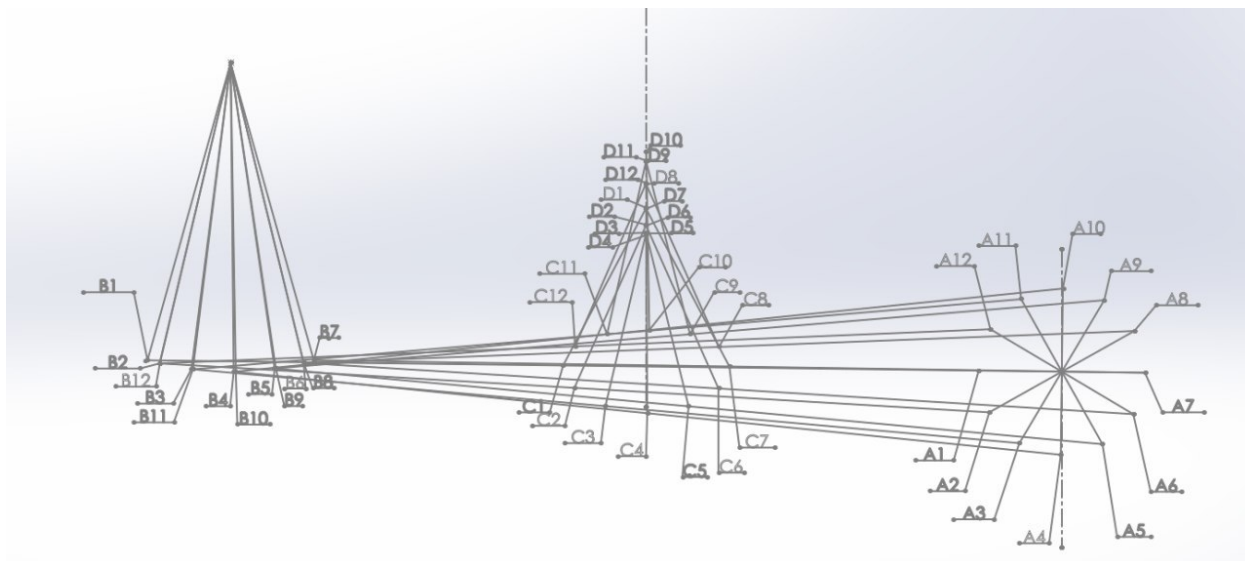


Рисунок 12 – 12 положень механізму

Висновок

Програмний метод побудови планів на мові Processing представляє собою комбінацію графічних і аналітичних методів з їх основними перевагами і недоліками. Головним недоліком програмного методу є складність алгоритму побудови планів положень, швидкостей та прискорень, а перевагою – такий підхід дозволяє студентам розвинути навички програмування при вирішенні задач механіки, а також одноразова розробка алгоритму побудови дозволить вирішувати подібні задачі за лічені секунди. Ще однією з переваг даного методу є можливість анімації руху плоского важільного механізму будь-якої складності.

Для побудови планів положень, швидкостей, прискорень та анімації руху ланок використовуються елементи векторної алгебри та аналітичної геометрії, а саме: пошук координат кінця відрізка за відомими координатами початку та значенням його довжини; пошук точок перетину двох кіл або кола з прямою (оскільки плани положень механізмів, які включають в себе двохповідкові групи, будуються методом засічок); знаходження рівняння прямої; визначення координат точки перетину двох прямих; тощо.

Реалізація анімації руху плоского важільного механізму в середовищі розробки Processing реалізована шляхом використання методу `void draw()`. За замовчуванням блок коду, який розташовано в даному методі буде безперервно виконуватися шістдесят разів в секунду. Саме тому швидкість анімації було зменшено для більш плавного відображення.

Враховуючи, що переваг у такого методу значно більше ніж недоліків, у даній роботі проведено перші ітерації процесу розробки універсального програмного забезпечення для кінематичного аналізу плоских важільних механізмів. Розроблено ряд методів для визначення кута обертання кривошипа, визначення положень шарнірів та ряд класів для створення ланок і стійок у вигляді об'єктів. Створено програмний код, який дозволяє автоматично будувати плани швидкостей та прискорень для будь-якого положення механізму.

У майбутньому планується розширити функціонал програмного забезпечення для реалізації можливості створення довільного плоского важільного механізму. Також планується вдосконалити програму таким чином, щоб процеси створення окремих ланок і механізму в цілому, а також, щоб побудова відповідних планів не потребувала специфічних знань теорії машин і механізмів від кінцевого користувача. Такий підхід дозволить значно розширити коло користувачів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Семеренко И. П., Катритсис Д. «Применение информационных технологий при преподавании технической механики Молодой ученый. — 2016. — №22.2. — С. 20-22.». [Электронный ресурс]: Режим доступа: <https://moluch.ru/archive/126/33662/>
2. Грінченко В. Т. «Вісник Київського національного університету імені Тараса Шевченка». [Электронный ресурс]: Режим доступа: irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/VKNU_fiz_mat_2013_3_6.pdf
3. Информационные технологии в решении прикладных задач механики. [Электронный ресурс]: Режим доступа: <http://innosfera.by/node/769>
4. Комп'ютер і механіка. [Электронный ресурс]: Режим доступа: irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/VKNU_fiz_mat_2013_3_6.pdf
5. Чертим от руки или все-таки на компьютере. [Электронный ресурс]: Режим доступа: http://www.ytchebnik.ru/program/publication_1940/
6. Система автоматизованого проектування і розрахунку. [Электронный ресурс]: Режим доступа: https://uk.wikipedia.org/wiki/Ситстема_автоматизованого_проектування_і_розрахунку#Функціональний_огляд_найпоширеніших_CAD-програм
7. Мова моделювання. [Электронный ресурс]: Режим доступа: https://uk.wikipedia.org/wiki/Мова_моделювання
8. Processing (programming language). [Электронный ресурс]: Режим доступа: [https://en.wikipedia.org/wiki/Processing_\(programming_language\)](https://en.wikipedia.org/wiki/Processing_(programming_language))
9. Processing for Programmers. [Электронный ресурс]: Режим доступа: <http://www.eliotlash.com/2011/08/processing-for-programmers/>

10. **Кинематический анализ плоских рычажных механизмов.**
[Электронный ресурс]: Режим доступа:
<https://studfiles.net/preview/5734283/page:5/>
11. П. Н. Сильченко, М. А. Мерко, М. В. Меснянкин, А. В. Колотов, Е. В. Беляков. «Теория механизмов и машин», 2008. – 56с.