МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2024 г.

Моделирование гидравлических процессов в петлевой системе охлаждения медных плит кристаллизатора машины непрерывного литья заготовок

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2024.406 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

Автор работы,
студент группы КЭ-406
_____Я.А. Усолкин
«___»_____2024 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2024 г.

Челябинск-2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-406
Усолкину Яну Александровичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

Тема работы: «Моделирование гидравлических процессов в петлевой системе охлаждения медных плит кристаллизатора машины непрерывного литья заготовок» утверждена приказом по университету от «__» _____ 2024 г. №

Срок сдачи студентом законченной работы: 01 июня 2024 г.

Исходные данные к работе: кристаллизатор машины непрерывного литья заготовок (МНЛЗ).

Диаметр вертикальных каналов d, мм	20;
Диаметр горизонтальных каналов D, мм	35;
Скорость воды в каналах Q_0, м^3/ч	17;
Число каналов в одной стенке, n	35;
Расстояние между вертикальными каналами $l_г$, мм	35;
Высота вертикальных каналов $l_в$, мм	1080;
Кинематическая вязкость воды ν, м^2 / с	$0{,}805 \cdot 10^{-6}$;

Эквивалентная шероховатость медных каналов $\Delta_э$, мм	0,01;
Расход жидкости G, кг/ч	12000;
Плотность воды при температуре 25 °С ρ, кг/м^3	998;
Динамический коэффициент вязкости воды μ, Па·с	$8{,}9 \cdot 10^{-4}$;
Длина канала для последовательного соединения L, м	55;
Диаметр канала для последовательного соединения $d_{расч}$, м	0,034.

Перечень подлежащих разработке вопросов:

1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.
2. Разработка математической модели МНЛЗ.
3. Разработка компьютерной модели МНЛЗ.
4. Проверка адекватности и достоверности созданных моделей.

Дата выдачи задания: ___ декабря 2023 г.

Руководитель работы _____ */ Д.В. Топольский /*

Студент _____*/ Я.А. Усолкин /*

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2024	
Аналитика проекта и его актуальность	01.04.2024	
Разработка математических моделей	15.04.2024	
Разработка компьютерных моделей	01.05.2024	
Проверка адекватности и достоверности созданных моделей	15.05.2024	
Компоновка текста работы и сдача на нормоконтроль	24.05.2024	
Подготовка презентации и доклада	30.05.2024	

Руководитель работы _____/ *Д.В. Топольский* /

Студент _____/ *Я.А. Усолкин* /

АННОТАЦИЯ

Я.А. Усолкин. Моделирование гидравлических процессов в петлевой системе охлаждения медных плит кристаллизатора машины непрерывного литья заготовок. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 69 с., 15 ил., 3 табл., библиогр. список – 21 наим.

В рамках выпускной квалификационной работы был проведен анализ математической модели, на основе которой разработана компьютерная программа на языке программирования C++, а также компьютерная модель на основе MATLAB с визуализацией расчетных данных гидравлической работы петлевой системе водяного охлаждения широких и узких стенок кристаллизатора машины непрерывного литья заготовок.

В ходе работы были проведен обзор научной и технической литературы, разработан алгоритм решения задач потери давления потока ламинарного или турбулентного потоков для последовательного соединения каналов, а также рассмотрена математическая модель для сложного соединения. На основе этих данных были разработаны и использованы компьютерные программы ConsoleInterface и Кристаллизатор-гидравлика для визуализации и гидравлической работы петлевой системе водяного охлаждения широких и узких стенок кристаллизатора машины непрерывного литья заготовок.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Непрерывная разливка металлов – это технология производства металлических слитков, при которой жидкий металл непрерывно заливается в форму (кристаллизатор) и одновременно вытягивается из нее в виде твердого слитка.

Непрерывная разливка является важным технологическим процессом в современном металлургическом производстве. Она позволяет производить металлические слитки с высокой производительностью, низкими затратами и повышенным качеством [1].

Современные металлургические технологии по получению черных и цветных металлов и их сплавов в виде слитков широко используют медные кристаллизаторы. От их работы зависит:

- производительность агрегатов;
- качество получаемых слитков.

Конструкция кристаллизатора должна обеспечивать:

- хороший теплоотвод;
- высокую химическую и механическую стойкость;
- безопасность работы.

1. Аналитика проекта

1.1. Потенциальные заказчики

В настоящее время, как в России, так и во всем мире, особое значение придается совершенствованию процесса непрерывной разливки стали и увеличению объемов стали, обрабатываемой на машинах непрерывной разливки, в основном за счет технологий, которые обеспечивают фазовый переход металла из жидкого состояния в твердое и удерживают слиток во время его вытягивания.

Основными заказчиками МНЛЗ являются металлургические комбинаты, например, как: магнитогорский (ПАО «ММК»), череповецкий (ОАО «Северсталь»), нижнетагильский (ОАО «Евраз») и другие.

1.2. Анализ предметной области

Во многих странах большая часть производимой стали разливается на машинах непрерывной разливки. Жидкая сталь непрерывно заливается в водоохлаждаемую форму, называемую кристаллизатором. Перед началом заливки в кристаллизатор помещается специальное устройство с зажимным захватом, которое служит основанием для первой порции металла. После затвердевания металла затравка вытягивается из кристаллизатора, увлекая за собой формирующийся слиток. Подача жидкого металла продолжается, и слиток непрерывно растет. В кристаллизаторе затвердевают только поверхностные слои металла, образуя твердую оболочку слитка, которая удерживает жидкую фазу вдоль центральной оси. Именно поэтому за кристаллизатором находится «вторая зона» (зона вторичного охлаждения). В этой зоне в результате поверхностного охлаждения происходит процесс кристаллизации заготовки. Этот процесс является способом получения слитков любой задуманной длины. [2]

1.3. Классификация МНЛЗ

Группируют МНЛЗ по [3]:

1 Геометрии кристаллизатора:

1.1 Вертикальные – это кристаллизаторы, имеющие вытянутую форму. В связи с этим подготовка и образование заготовки происходят в данной плоскости (вертикальной), как показано на рисунке 1;

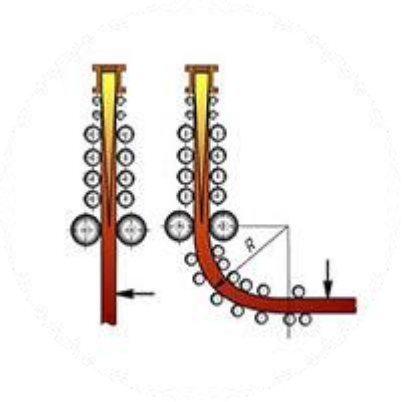


Рисунок 1 – Вертикальный кристаллизатор (слева) и вертикальный с загибом (справа) [4]

1.2 Радиальные – конструктивной особенностью данного типа машин является наличие кристаллизатора с конкретно заданным радиусом, благодаря которому происходит получение радиальной технологической линии (рисунок 2);

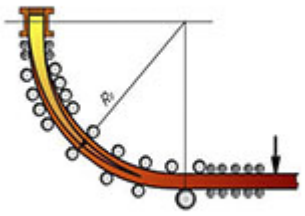


Рисунок 2 – Радиальный кристаллизатор

1.3 Криволинейные – имеет изогнутую форму, по траектории которой

движется слиток. Это способствует равномерному распределению тепла и, соответственно, более равномерному затвердеванию, что снижает вероятность возникновения внутренних дефектов (рисунок 3);

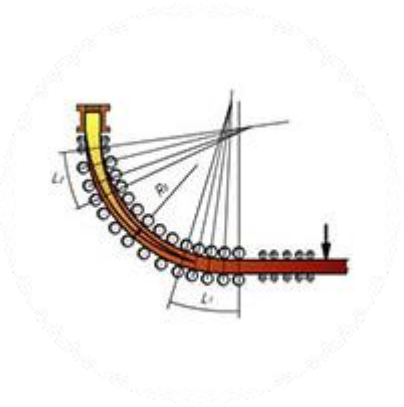


Рисунок 3 – Криволинейный кристаллизатор

1.4 Горизонтальные – позволяют производить слитки больших размеров. Плюсами таких кристаллизаторов являются: компактность, увеличенная производительность, улучшенное качество поверхности. Однако, достаточно дорогие в обслуживании (рисунок 4).

Рисунок 4 – Горизонтальный кристаллизатор

2 Геометрии слитка:

2.1 Слябовые – это полупродукты металлургического производства, имеющие форму плиты или пластины (рисунок 5). Отличительная особенность данного вида слитков заключается в большом отношении ширины к высоте прямоугольного сечения (ширина сляба от 400 до 2500 мм, а высота (или же толщина) от 75 до 600 мм);

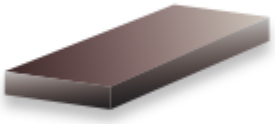
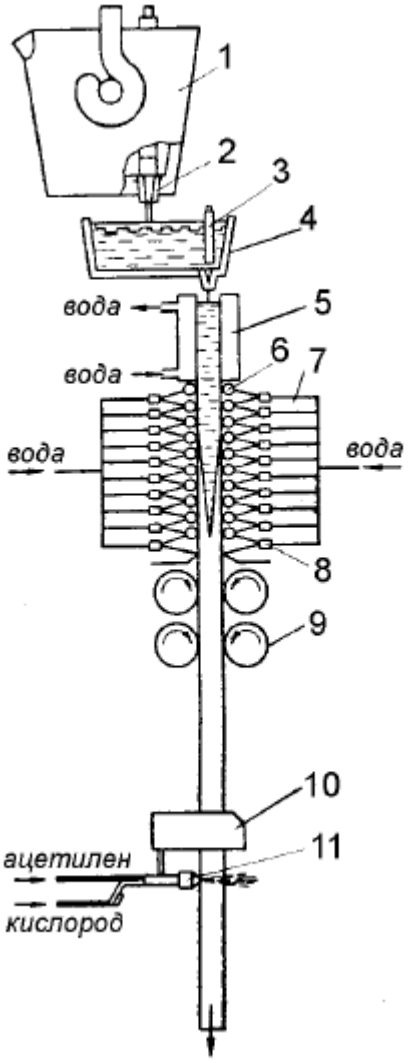


Рисунок 5 – Слябовый слиток

2.2 Блюмовые (или же “обжатая болванка” [5]) – это слитки, стальная заготовка сечения которых близка к квадратному. Соотношение ширины к высоте могут разниться от 140х140 мм до 450х450 мм. Являются аналогичными заготовками по отношению к слябовым слиткам (рисунок 6).

Рисунок 6 – Блюмовый слиток

Наибольшее распространение получили машины непрерывного литья заготовок (МНЛЗ) вертикального типа (рисунок 1, рисунок 7), подробнее на нижеприведенной схеме:



Условные обозначения:

1 – сталеразливочный ковш; 2 – шиберный затвор; 3 – стопорный затвор; 4 – промежуточный ковш; 5 – кристаллизатор; 6 – опорные валки; 7 – зона вторичного охлаждения; 8 – форсунки для распыления воды; 9 – тянущие валки; 10 – синхронизатор перемещения газорезки со скоростью заготовки; 11 – газорезка.

Рисунок 7 – Схема МНЛЗ вертикального типа [6]

Непрерывная разливка на машинах непрерывного литья (МНЛЗ) использует сталеразливочные ковши. Расплавленный металл из ковша направляется в водоохлаждаемый кристаллизатор через специальное промежуточное устройство, которое предотвращает попадание шлака в кристаллизатор и обеспечивает точную регулировку скорости разливки. Непрерывность процесса гарантирует равномерную структуру слитка по всей

длине, что достигается за счет непрерывного литья и кристаллизации.

В промежуточном ковше для каждого кристаллизатора устанавливаются механизмы регулировки скорости разливки. Количество кристаллизаторов зависит от требуемой производительности, которая, в свою очередь, определяет количество ручьев разливки на МНЛЗ.

Конструкция и режим зоны вторичного охлаждения МНЛЗ должны оптимизировать процесс затвердевания слитка, обеспечивая быстрое застывание и медленное охлаждение для предотвращения образования трещин.

Роликофорсуночное охлаждение, наиболее распространенный метод, использует форсунки для опрыскивания водой и ролики для равномерного распределения воды по поверхности слитка, предотвращая деформацию краев при разливке крупных слитков.

Для оптимизации расхода воды, зону вторичного охлаждения МНЛЗ делят на несколько секций.

По мере опускания полностью затвердевшего нижнего конца слитка, от него отрезаются заготовки заданной длины. Отделенные заготовки отправляются в следующую зону кристаллизации для окончательной застывания.

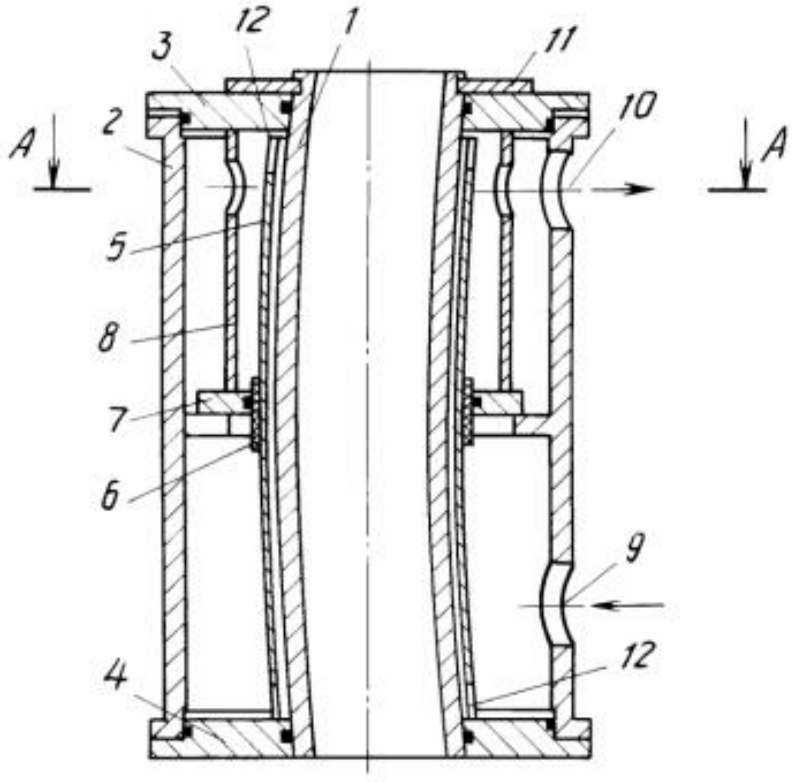
В зависимости от радиуса величины закругления, определяется и высота МНЛЗ. При расчёте радиуса закругления, а также клетей вторичного охлаждения учитывают, чтобы к выходу в горизонтальное положение слиток был абсолютно застывшим.

1.4. Типы кристаллизаторов МНЛЗ

Кристаллизаторы используются для обеспечения начальной кристаллизации и формирования стальных слитков. Наиболее распространенным типом кристаллизатора является сборный кристаллизатор.

Стенки формы могут быть прямыми или волнистыми. Стенки обычно параллельны друг другу по высоте формы. При заливке плоских бревен

большого сечения стенки обычно сужаются назад на 1%, так как при усадке бревно отходит от стенок, и теплоотдача в этой зоне значительно снижается. Кристаллизатор представлен на рисунке 8.



Условные обозначения:

1 - медная гильза; 2 – корпус; 3 - верхняя уплотнительная крышка, 4 – нижняя уплотнительная крышка, 5 - циркуляционная обойма; 6 - нанесенный полимерный пояс; 7 - уплотнительная диафрагма, 8 - нажимная втулка; 9 - подвод воды; 10 - отвод воды; 11- шпонка; 12 - прорези для входа и выхода охладителя.

Рисунок 8 – Кристаллизатор МНЛЗ [7]

Высота кристаллизатора МНЛЗ должна обеспечивать образование достаточно толстой корки в заготовке при выходе, исключающей возможность ее прорывания.

Для предотвращения ослабления заготовки формы настроены на возвратно-поступательное движение (колебание). Режим работы стенда обеспечивается применением электрических цепей, рычагов и гидравлической системой колебания формы.

Для уменьшения трения между стенками формы МНЛЗ и поверхностью заготовки во время литья форма обычно смазывается автоматически.

Смазка подается либо через канавку выше металлического мениска, либо в верхнюю часть формы. В качестве смазки используются парафин, репейник, репейное и рапсовое масло.

Основными причинами выхода из строя кристаллизатора являются износ и деформация медных стенок, раскрытие стыка между стенками и потеря меди в стыке. Для восстановления медных стенок обычно используется перешлифовка.

1.5. Функциональные требования

1. Проектирование и усовершенствование комплекса МНЛЗ, а также отдельных узлов и систем.
2. Производство кристаллизаторов с высокопрочным покрытием.
3. Гидравлический расчет кристаллизаторов.
4. Производство секций вторичного охлаждения.
5. Выявление проблем.

1.6. Нефункциональные требования

1. Система диагностики температурных сенсоров кристаллизатора.
2. Производительность МНЛЗ.
3. Удобство сопровождения.
4. Расширяемость и надежность.
5. Факторы эксплуатации.

1.7. Анализ литературы

Целью данного анализа является выяснение используемых принципов и методов по изучению непрерывной разливки металлов, а также сравнение разработанных цифровых, компьютерных, математических моделей среди зарубежной и отечественной литературы.

1.7.1. Зарубежная литература

Brian G. Thomas, Modeling of the Continuous Casting of Steel—Past, Present, and Future [8].

В данной статье Б. Томаса было показано несколько примеров применения математических моделей для количественной оценки и понимания механизмов образования дефектов при непрерывной разливке сляба. Выводом данной работы является рассуждение о необходимости упрощения существующих математических моделей, упрощения ключевых явлений прежде, чем они смогут достичь улучшения своего полного потенциала и полезности. Чтобы эти процессы процветали, а сталь была качественнее, кривая обучения оптимизации процессов должна производить ее короче.

YA MENG and BRIAN G. THOMAS, Heat-Transfer and Solidification Model of Continuous Slab Casting: CON1D [9].

Рассматривается простая, но комплексная модель теплового потока. Для нее разработана форма кристаллизатора для отливки стали с зазором и оболочкой. В данной модели происходит одномерный процесс затвердевания стальной оболочки, а также двумерный процесс рассеивания тепла движущегося твердого тела и жидкого шлака в межфазном зазоре. Идет расчет теплопроводимости внутри медной стенки кристаллизатора. Выводом работы является удобство использования данной модели на персональном компьютере. Это было подтверждено посредством числовых сравнений и калибровки с измерениями на работающих литейных машинах, включая повышение температуры охлаждающей жидкости, толщину оболочки,

толщину слоя шлака и термопары. Помимо теплопередачи, модель предсказывает толщину затвердевших слоев шлака, идеальную конусность формы и выясняет потенциальные проблемы с качеством, такие как затвердевание и кипение шлака в водных каналах

J. Yang, Z. Xie, Z. Hu, H. Meng, A Three-Dimensional Real-time Heat Transfer Model for Continuous Casting Blooms [10].

В этой статье была разработана трехмерная модель теплопередачи в реальном времени для непрерывной разливки блюмов. Из данного исследования можно сделать следующие выводы: данная модель позволяет рассчитывать весь процесс разливки, однако, если начальная разливка проходит в трехмерной модели, то стационарный этап может моделироваться лишь в двумерной. Также представлен алгоритм решения моделей в реальном времени, который удовлетворяет требованиям как эффективности, так и численной точности. За счет использования наследуемой переменной неравномерной сетки и переменных шагов, расчет можно ускорить от 110 до 200 раз. Помимо блюмов, 3D-модель в реальном времени также можно применять к слябовым заготовкам, поскольку все они имеют прямоугольную форму и схожесть друг с другом. Среди них разливочная машина для заготовок является самой сложной в режиме реального времени из-за необходимости крайне маленького по времени шага.

1.7.2. Отечественная литература

Гусев М.П., Анисимов К.Н., Зарубин С.В., Лонгинов А.М., Концепт цифрового двойника процесса непрерывного литья слитков [11].

В данной статье предлагается внедрение на металлургические предприятия цифрового двойника слитка. Это обусловлено тем, что он позволит решить такие задачи, как: повышение качества непрерывнолитого слитка и, как следствие, повышение рентабельности производства. Изначально цифровой двойник будет нести в себе рекомендательный характер, но после некоторого времени эксплуатации будет предлагаться

автоматическое изменение параметров. Также внедрение системы динамического охлаждения вторичной зоной, причем изменение расходов в зонах будет связано с граничными условиями в связанной тепловой и деформационной задачах и с качеством слитка. Внедрение системы мягкого обжатия слитка, которая также будет связана с математической моделью слитка и с его качеством. Внедрение в кристаллизатор оптоволоконной системы измерения температур, позволяющей в режиме реального времени определить распределение температур и теплового потока в кристаллизаторе и снижающую эксплуатационные расходы на смену температурных датчиков, поскольку надежность оптоволоконных измерителей существенно выше, чем используемых в настоящее время классических термопар.

Т.П. Ларина, К.Н. Вдовин, И.М. Ячиков, Расчет гидравлических параметров медных кристаллизаторов со сложным или последовательным соединением каналов [12].

В данной статье разработан алгоритм гидравлического расчета стенки кристаллизатора, имеющей произвольное число горизонтальных и вертикальных каналов. Создана компьютерная программа, которая может быть полезна при конструировании новых медных кристаллизаторов для выбора наиболее рациональных режимов их тепловой и гидродинамической работы, а также при реконструкции уже существующих кристаллизаторов с целью перехода от конструкции стенок со сверленными к щелевым охлаждающим каналам.

Вывод по первой главе

Проведена аналитика проекта, итогами которой стала ясна предметная область, а также определены основные и потенциальные заказчики машин непрерывной разливки стали.

В результате анализа предметной области и существующих статей и моделей из зарубежных и отечественных источников был выбран метод реализации поставленной задачи.

Если среди иностранных специалистов по большей части рассматривается двумерная и трехмерные модели, то исследователи из Российской Федерации упор ставят на математическую, трехмерную модели, а также на цифровые двойники.

В качестве основы была выбрана разработка компьютерной модели на основе математической модели Т.П. Лариной, К.Н. Вдовина, И.М. Ячикова, так как проектируемые задачи были поставлены для магнитогорского металлургического комбината (ПАО «ММК»), который является основным потенциальным заказчиком

2. Математическая модель

2.1. Определение потери давления потока жидкости

Теплопередача через стенки медного кристаллизатора осуществляется посредством воды, протекающей по системе каналов. Для каждой стенки кристаллизатора может применяться как схема прямоточного движения воды в каналах, когда вода подается сверху вниз, т.е. в направлении движения непрерывного слитка, так и противоточного движения с подачей воды снизу вверх, т.е. в направлении, противоположном движению фронта кристаллизации.

Потери напора, обусловленные большими гидравлическими сопротивлениями в системе охлаждения могут привести к снижению давления, т.е. падению температуры на линии насыщения, возникновению нежелательного местного кипения воды и ухудшению тепловой работы кристаллизатора [13-16].

Цель работы состоит в создании математической модели гидравлического расчета стенки кристаллизатора.

Первичная система охлаждения МНЛЗ представляет собой сложный трубопровод с N-вертикальными каналами (рисунок 9).

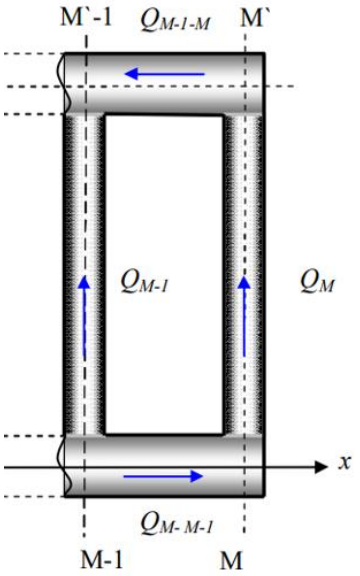


Рисунок 9 – Схема гидравлической работы стенки кристаллизатора [14]

2.2. Схема взаимодействия для сложного соединения

Данная схема вертикальной стенки кристаллизатора будет использована для решения задачи. В первом случае, неизвестным будет являться расход жидкости с заданным перепадом давления, во втором же, неизвестен перепад давления с заданным расходом жидкости. Основная задача – нахождение зависимости перепадов давления.

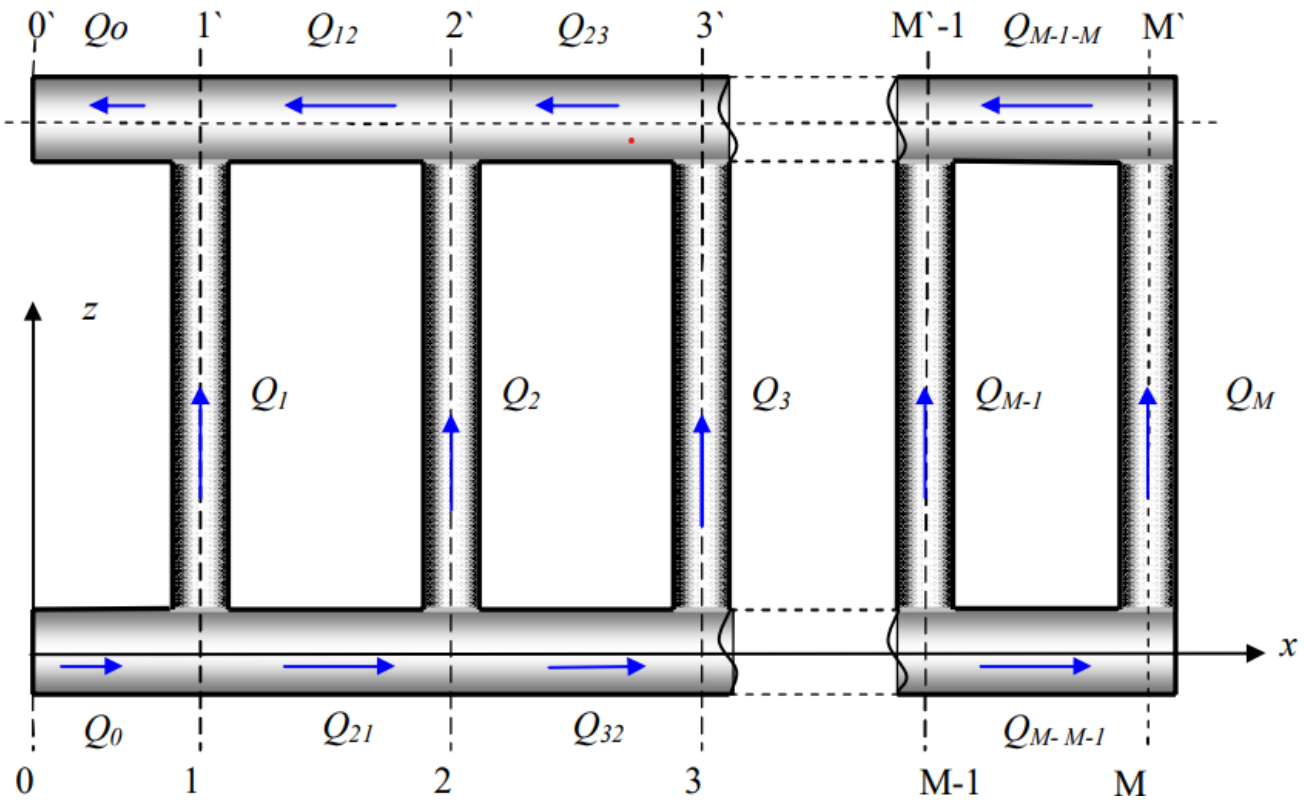


Рисунок 10 – Схема гидравлической работы стенки кристаллизатора со сложным соединением при подаче воды снизу [17]

2.3. Математическая модель для сложного соединения

Обозначим площади сечения горизонтальных и вертикальных каналов соответственно F и f. Пусть известны длины вертикального и горизонтального каналов, а также их эквивалентные диаметры – d и D соответственно. Для произвольного количества M вертикальных каналов имеем $2M$ особых точек гидравлической системы, где наблюдается местные сопротивления, эти точки будем обозначать как 1, 1', 2, 2' и т.д. (рисунок 10).

Данная гидравлическая система имеет $L = 2M - 2$ узлов, так как самый правый вертикальный канал не образует разветвление потока воды. В данной постановке неизвестными являются расходы воды, проходящей в M вертикальных и $M - 2$ горизонтальных каналах всего $2M - 2$ расхода. Неизвестными являются давления во всех узлах и на выходе из кристаллизатора, всего $L + 1$ давление. Общее число неизвестных $X = L + 1 + 2M - 2 = 2L + 1$ или в зависимости от числа вертикальных каналов $X = 4M - 3$

При разработке математической модели для определения неизвестных расходов жидкости в горизонтальных и вертикальных каналах использовали условия балансов расходов воды и гидравлических напоров $\left(Q_{j,j-1} - Q_j - Q_{j+1,j} = 0\right)$. Предполагаем, что поток жидкости идеальный, поэтому применим уравнение Бернулли:

$$z_1+\frac{p_1}{\rho\cdot g}+\frac{\alpha_1 v_1^2}{2g}=z_2+\frac{p_2}{\rho\cdot g}+\frac{\alpha_2 v_2^2}{2g}+h_\omega \quad , \qquad (1)$$

где z_1 и z_2 – удельные энергии положения, характеризующие потенциальную энергию в сечениях 1–1 и 2–2;

$\frac{p_1}{\rho g}$ и $\frac{p_2}{\rho g}$ – удельные энергии давления, характеризующие потенциальную энергию давления в тех же сечениях;

$\frac{v_2^1}{2g}$ и $\frac{v_2^2}{2g}$ – удельные кинетические энергии в тех же сечениях.

Составлены дополнительные уравнения на основе баланса напоров для

M – 1 контуров. Обход производится по направлению против хода часовой стрелки для произвольного контура:

$$h_n(U_{i-1}-U_i)+2h_г(U_i)+h_б(U_i-U_{i+1},U_i)+h_в(U_i-U_{i+1})+ \\ +h'_б(U_i-U_{i+1},U_i)+h'_n(U_{i-1}-U_i,U_{i-1})-h'_б(U_{i-1}-U_i,U_{i-1})- \\ -h_в(U_{i-1}-U_i)-h_б(U_{i-1}-U_i,U_{i-1})=0. \quad , \qquad (2)$$

где $h_n(U_i)$, $h_г(U_i)$, $h_б(U_i)$, $h_в(U_i)$ – потери напора по длине в i-ых вертикальных и горизонтальных каналах.

В случае, если известен или задан перепад давления воды ΔP между нулевыми сечениями, тогда систему можно решить следующим образом:

$$(\Delta P/\gamma)-2h_г(U_0)-h_{16}(U_0-U_1,U_0)-h_1(U_0-U_1)-h'_{16}(U_0-U_1,U_0)=0, \\ U_M=0, \quad ; \qquad (3)$$

где $\gamma = \rho \cdot g$, ρ – плотность воды

2.4. Схема взаимодействия для последовательного соединения.

Данная схема трубопровода будет использована для решения задачи со следующими неизвестными: скорость движения потока, число Рейнольдса, тип потока (ламинарный/турбулентный), потеря давления потока.

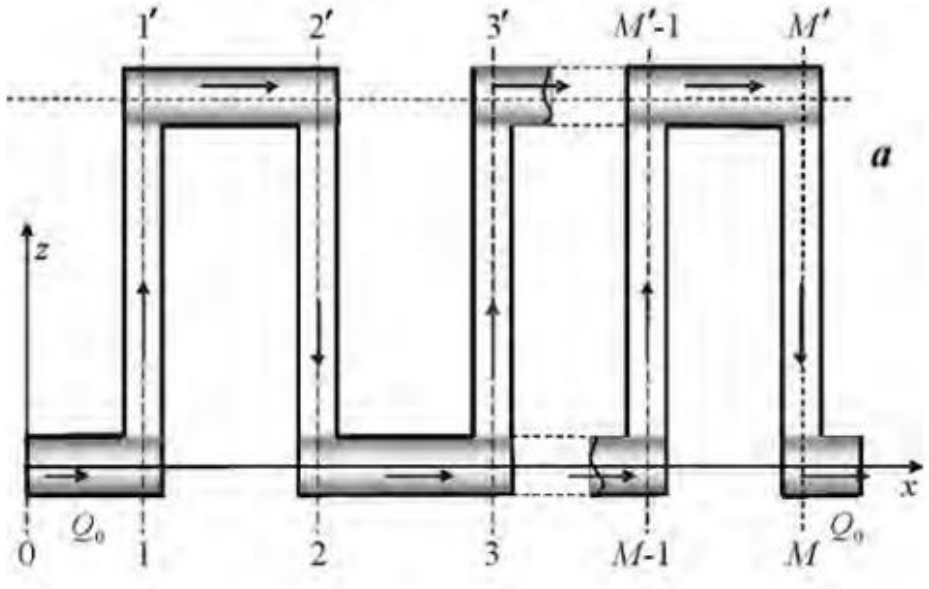


Рисунок 11 – схема гидравлической работы стенки кристаллизатора

а – последовательное соединение каналов [18]

2.5. Математическая модель для последовательного соединения

Для расчета неизвестных расходов жидкости применяли законы Кирхгофа. Для узлов j = 1, 2, ..., M – 1 записывали уравнения на основе условия баланса расходов:

$$Q_{j,j-1} - Q_j - Q_{j+1,j} = 0\ ; \qquad (4)$$

где Q_{ij} – расход воды на участке между i и j вертикальными каналами.

Создав модель перепада давления и расхода жидкости на участке трубопровода с двумя коленами, можно масштабировать данную модель на M-колен с учетом значений давления и расхода воды в узлах, согласно закону Кирхгофа и условию баланса (1).

Рассмотрим гидравлически гладкую трубу длину которой составляют два горизонтальных участка длиной l_h и участок вертикального канала длиной l_v. Тогда длина модельного участка трубопровода обозначим как L и будем искать по формуле:

$$L = 2l_h + l_v \qquad (5)$$

Для труб некруглого сечения в качестве диаметра трубы используют эквивалентный диаметр $d_э = 4f/p$, где f — площадь живого сечения, p — периметр канала, для труб круглой формы используем известное значение диаметра *d*.

Найдем потерю давления при перемещении охлаждающей жидкости с температурой 25 ^{0}C в количестве 12 т/ч.

1. Составляем таблицу исходных данных:

G = 12 т/ч = 12 000 кг/ч;

$G_c = \frac{12000}{3600}$ кг/с;

ρ = 998 кг/м^3;

μ = 8,9·10^{-4} Па·с;

L = 55 м;

$d_{расч}$ = 0,034 м.

Таблица 1 – Местные сопротивления

Вид сопротивления	Число сопротивлений	Коэффициент сопротивления ξ
Вход	1	0,5
Выход	1	1
Нормальный вентиль	1	6
Задвижка	1	0,5
Колено	2	1,8

Значения коэффициентов сопротивления, требуемые для расчета, обозначены в таблице Б.1 (Приложение Б) [19–21].

Определить: $\Delta p = ?$

Расчетная схема трубопровода приведена на рисунке 11. Для количества колен взято произвольное значение в виде двух (первые два изгиба трубопровода)

Схема решения задачи:

Основной задачей будет нахождение потери давления потока, для этого приведем формулу к общему виду, а после используем все заданные значения для расчетной формулы.

Общий вид:

$$\Delta p = \Delta p_{ск} + \Delta p_{тр} + \sum \Delta p_{сопр}; \quad (6)$$

Расчетная формула:

$$\Delta p = \frac{\omega^2 \cdot p}{2} \cdot \left(1 + \lambda \cdot \frac{L}{d} + \sum \xi\right), \text{ Па}; \quad (7)$$

Исходя из расчетной формулы, требуется найти неизвестные параметры в виде секундного объемного расхода и средней скорости. Помимо этого, необходимо понять тип потока и установить значение коэффициента внешнего трения. Для этого требуется найти число Рейнольдса:

- секундный объемный расход:

$$Q_c = \frac{G_c}{p}\, м^3/с;\ \ Q_c = \frac{G}{3600 \cdot p}\, м^3/с; \quad (8)$$

- средняя скорость:

$$\omega_{Cp} = \frac{Q_c}{F} = \frac{4 \cdot Q_c}{\pi \cdot d^2}\, м/с, \quad (9)$$

значение коэффициента внешнего трения в зависимости от режима движения, т.е. $\lambda = f(Re)$, для этого определим режим движения жидкости по числу Рейнольдса:

$$Re = \frac{\omega_{cp} \cdot d \cdot p}{\mu}. \quad (10)$$

Причем если Re <2320, то наблюдается ламинарный режим $\left(\lambda = \frac{64}{Re}\right)$, если Re >2320, то наблюдается турбулентный режим $\left(\lambda = \frac{0{,}316}{\sqrt[4]{Re}}\right)$.

Для расчета также потребуется сумма коэффициентов местных сопротивлений $\Sigma\xi$:

$$\Sigma\xi = \xi_1 + \xi_2 + \xi_3 + \xi_4 + 2 \cdot \xi_5; \quad (11)$$

Составим блок-схему решения данной задачи:

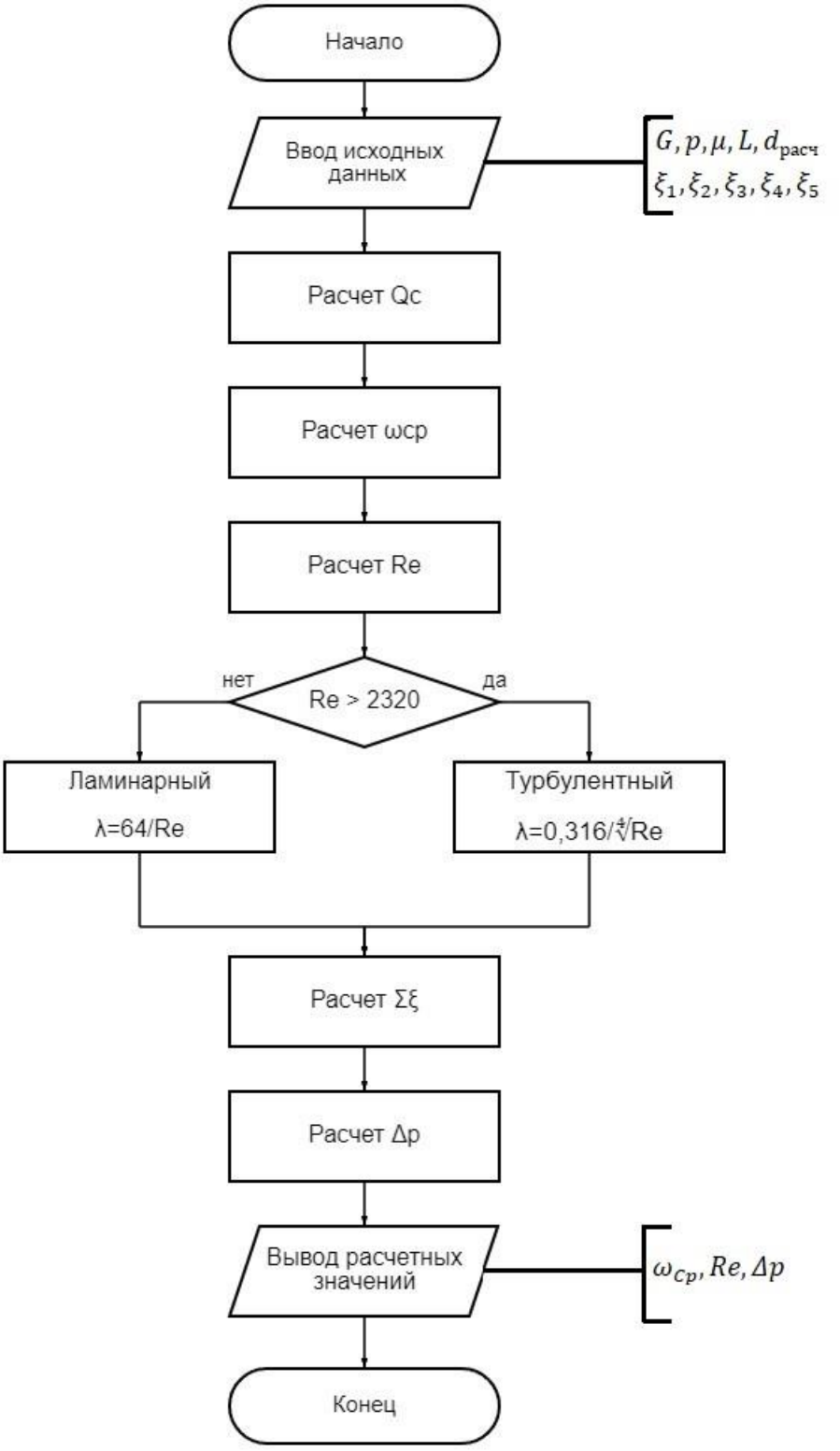


Рисунок 12 – Алгоритм решения задачи поиска потерь давления ламинарного или турбулентного потоков

Вывод по второй главе

В данной главе был описан алгоритм решения задачи моделирования гидравлических процессов на основе математической модели.

Основной задачей алгоритма при сложном соединении получить зависимости перепада давления, зависящего не только от расхода воды, а также от геометрических характеристик (в нашем случае, диаметры каналов, их длина и шероховатость).

Результатами расчета для потока с последовательным соединением каналов по данному алгоритму являются: скорость движения ω_{Cp} = 3,67 м/с; число Рейнольдса Re = 140256, благодаря этому определяем, что установленный режим является турбулентным, потеря давления потока Δp = 262478,5 Па.

На основе этих данных будет разработана компьютерная модель для расчета скорости движения, числа Рейнольдса, режима течений и потерь давления потока.

3. Компьютерное моделирование

С использованием математической модели Т.П. Лариной, К.Н. Вдовина, И.М. Ячикова, была реализована компьютерная программа, в которой проведены расчеты гидравлической работы системы охлаждения.

3.1. Диаграмма вариантов использования

Для проектирования модели был использован язык графического описания UML (англ. "Unified Modeling Language") – стандартизированный язык моделирования при проектировании программ. В соответствии с требованиями была построена диаграмма вариантов использования – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен пользователям (рисунок 13). В нашем случае диаграмма отражает модель взаимодействия актера «Пользователь» с системой.

Ввод набора данных для модели ConsoleInterface подразумевает под собой возможность ручного набора, а также возможность загрузки заданных значений с датасетов (в нашем случае был опробован датасет «Kaggle»)

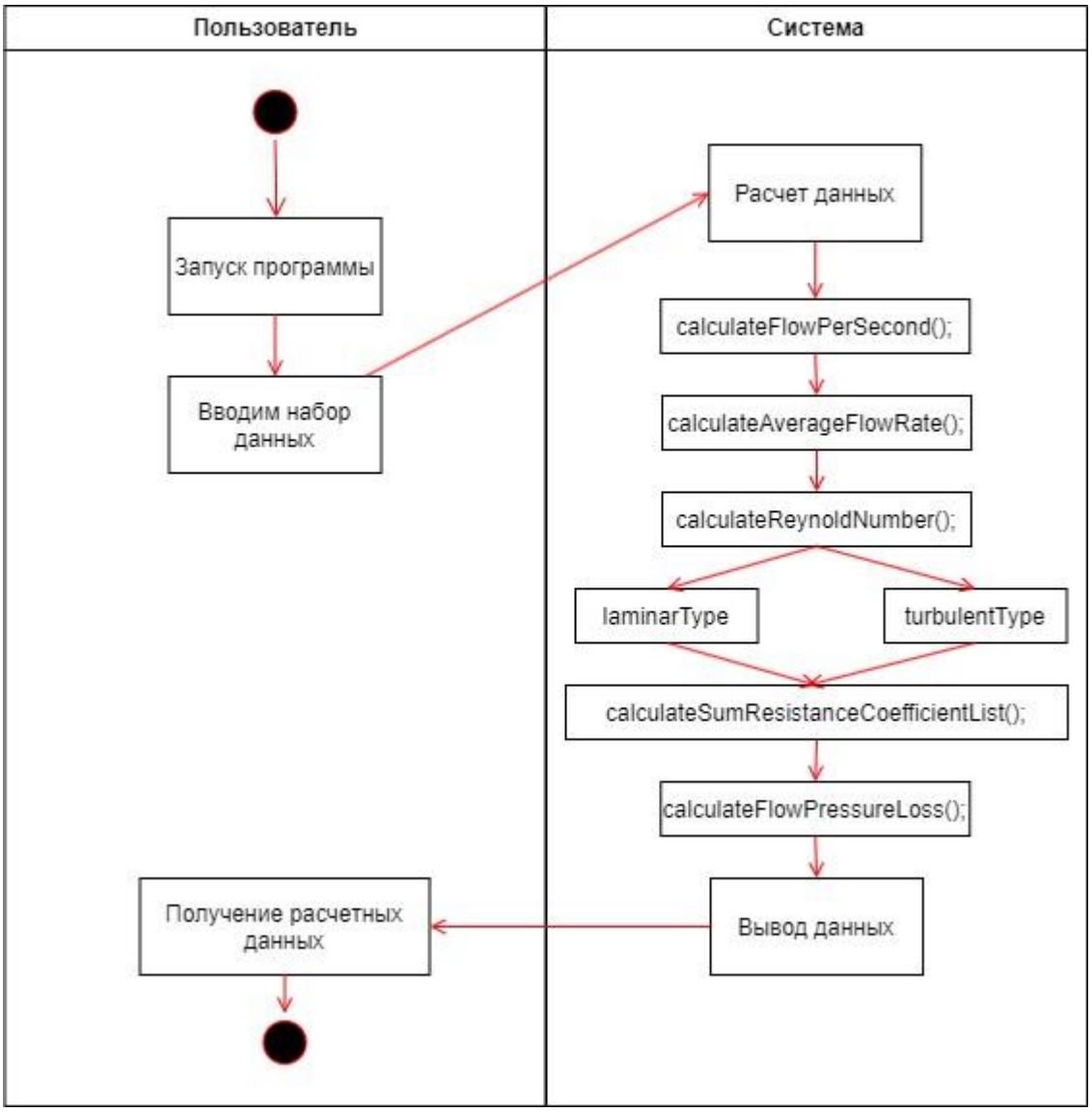


Рисунок 13 – Диаграмма деятельности

Проектируемая модель, реализованная как компьютерная программа на основе математической модели, занимается задачами определения потери давления потока жидкости для последовательного соединения каналов кристаллизатора и расчетом перепадов давления при подаче воды для сложного соединения каналов кристаллизатора.

Архитектура модели определяет ее компоненты, их функции и взаимодействие. Для представления компонентов модели была построена диаграмма компонентов. В ней реализовано выявление связи между всеми структурными компонентами. Разработанная диаграмма представлена на

рисунке 14.

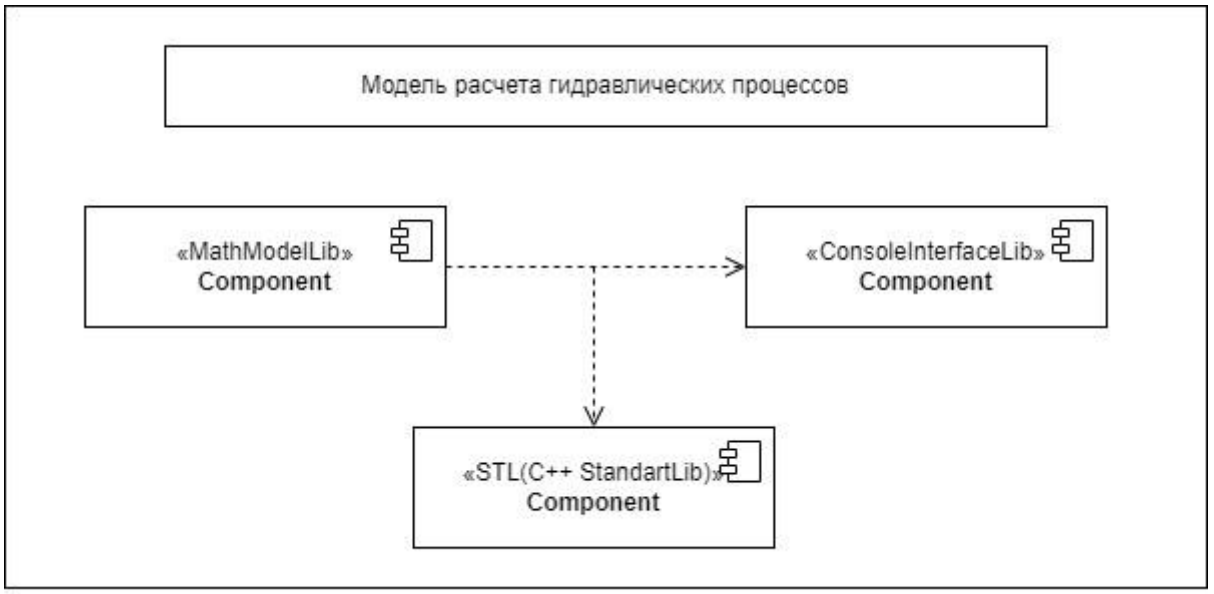


Рисунок 14 – Диаграмма компонентов

Данная диаграмма включает в себя такие библиотеки, как:

1. MathModelLib – основная разработанная библиотека, в которой реализована математическая модель, представленная во второй главе, целью которой является расчет данных по заданным параметрам для гидравлической работы.
2. ConsoleInterfaceLib – библиотека предоставляющая консольный интерфейс для взаимодействия с математической моделью (MathModelLib), позволяет пользователю вводить и редактировать данные для математической модели и передавать в нее же, а также выводить результат
3. STL (C++ StandartLib) – стандартная библиотека, позволяющая пользоваться функционалом C++.

3.2. Программирование для последовательного соединения

Разработаем программу по заданному алгоритму, описанному на рисунке 12, основной задачей является нахождение скорости движения потока, числа Рейнольдса и потери давления потока с минимальной погрешностью относительно математической модели.

Производится расчет параметров.

На листинге 1 представлен расчет секундного объемного расхода жидкости и средней скорости потока:

```
void MathModel::calculateVolumeFlowPerSecond() {
    m_volumeFlowPerSecond = m_params.consumptionPerHour /
(m_params.compressStress * 3600);
}

void MathModel::calculateAverageFlowRate() {
    m_averageFlowRate = 4 * m_volumeFlowPerSecond / (M_PI *
std::pow(m_params.crystallizerChannelDiameter, 2));

}
```

Листинг 1 – Расчет секундного объемного расхода жидкости и средней скорости потока

На листинге 2 отображен расчет числа Рейнольдса от которого зависит, какого типа будет поток (ламинарный или турбулентный):

```
void MathModel::calculateReynoldNumber() {
    m_reynoldNumber =
            (m_averageFlowRate * m_params.crystallizerChannelDiameter *
m_params.compressStress) / m_params.viscosity;
    if (m_reynoldNumber < 2320) {
        calculateLambdaForLaminar();
    } else {
        calculateLambdaForTurbulent();
    }

}
```

Листинг 2 – Расчет числа Рейнольдса и определение типа потока

На листинге 3 получение результата расчета значения коэффициента внешнего трения, в зависимости от типа потока:

```
void MathModel::calculateLambdaForLaminar() {
    m_lambda = 64 / m_reynoldNumber;
}

void MathModel::calculateLambdaForTurbulent() {
    m_lambda = 0.316 / std::pow(m_reynoldNumber, -0.25);
}
```

Листинг 3 – Расчет значения коэффициента внешнего трения

По итогам выполнения алгоритмов, описанных выше, можно произвести расчет потери давления потока. Алгоритм расчета представлен на листинге 4:

```
void MathModel::calculateFlowPressureLoss() {
    m_flowPressureLoss = 0.5 * std::pow(m_averageFlowRate, 2) *
m_params.compressStress *
                         (1 + (m_lambda * m_params.crystallizerLength * /
                               m_params.crystallizerChannelDiameter) +
                          m_sumResistanceCoefficientList);
}
```

Листинг 4 – Расчет значения потери давления потока

Таким образом, по разработанному в математической модели алгоритму гидравлического расчета стенки кристаллизатора, создана программа, позволяющая производить подсчеты по заданным параметрам для схемы взаимодействия с последовательным соединением каналов

3.3. Программирование для сложного соединения

Для расчета неизвестных расходов жидкости проще воспользоваться компьютерной программой “Кристаллизатор-гидравлика”. Она позволяет проводить гидравлический расчет плиты кристаллизатора, имеющей произвольное количество каналов различной формы и размеров. При этом пользователь может легко вводить и редактировать исходные данные, которые представляются в виде наглядных рисунков, схем и чертежей. Результаты компьютерного моделирования можно получать в виде таблиц, графиков и диаграмм.

С использованием программы можно проводить моделирование гидравлической работы системы охлаждения медной стенки, имеющей три вертикальных цилиндрических канала. Основные исходные данные для моделирования сведем в таблицу 2 для наглядности.

Таблица 2 – Исходные данные для гидравлического расчета кристаллизатора с тремя вертикальными каналами

Параметр	Обозначение	Единица измерения	Значение
Диаметр вертикальных каналов	d	мм	20
Диаметр горизонтальных каналов	D	мм	35
Высота вертикальных каналов	$l_в$	мм	1080
Расстояние между вертикальными каналами	$l_г$	мм	35
Кинематическая вязкость воды	ν	$м^2 / с$	$0{,}805 \cdot 10^{-6}$
Эквивалентная шероховатость медных каналов	$\Delta_э$	мм	0,01

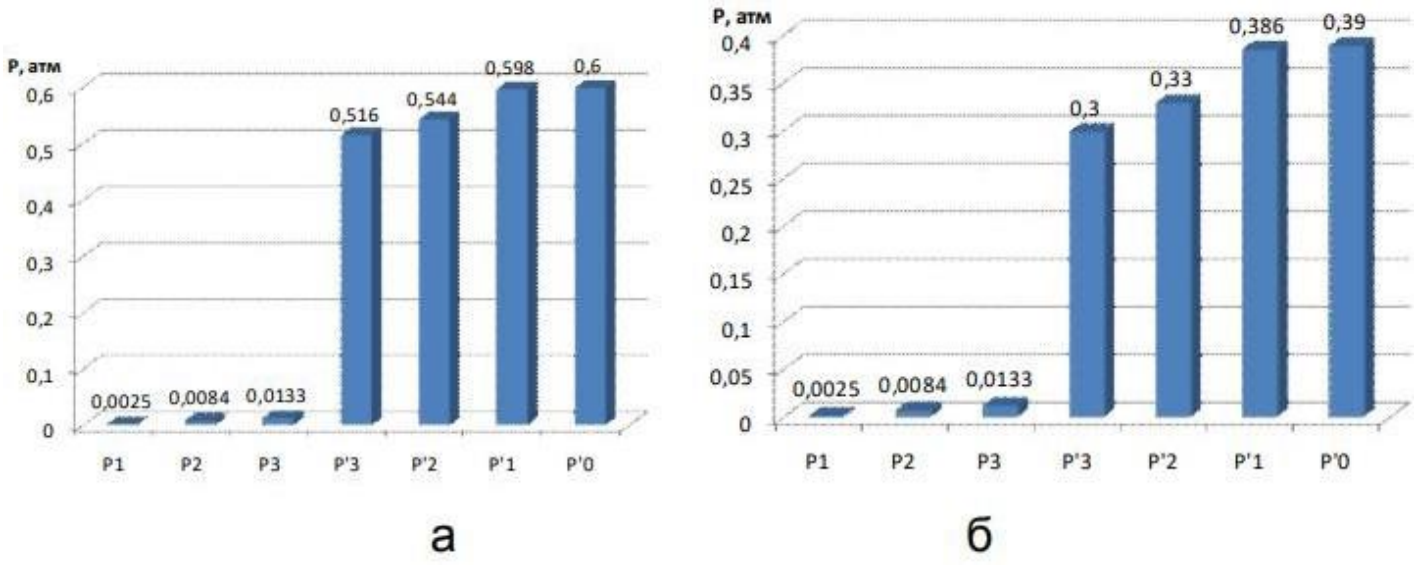


Рисунок 13 – Перепады давления в узлах относительно входного давления при подаче воды с расходом Q_0 = 17 м^3/ч: а – снизу; б –сверху

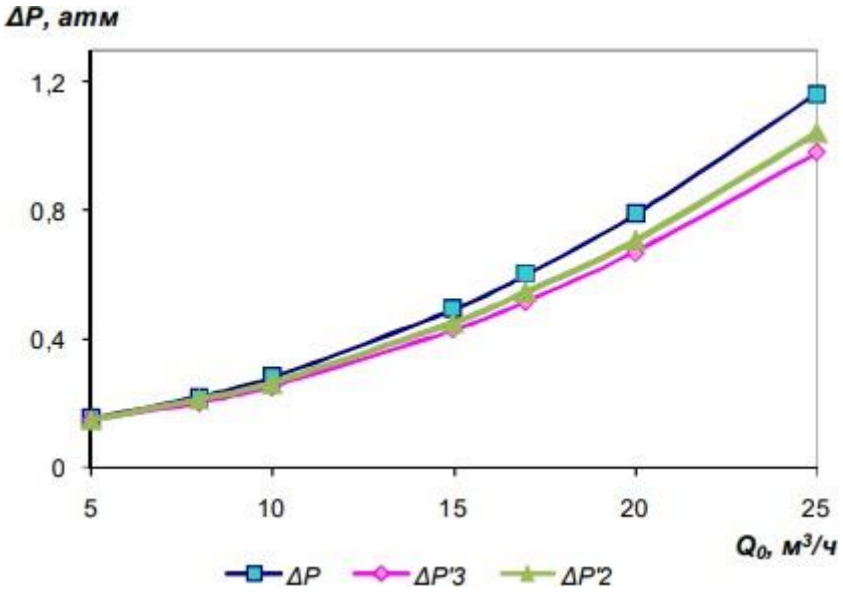
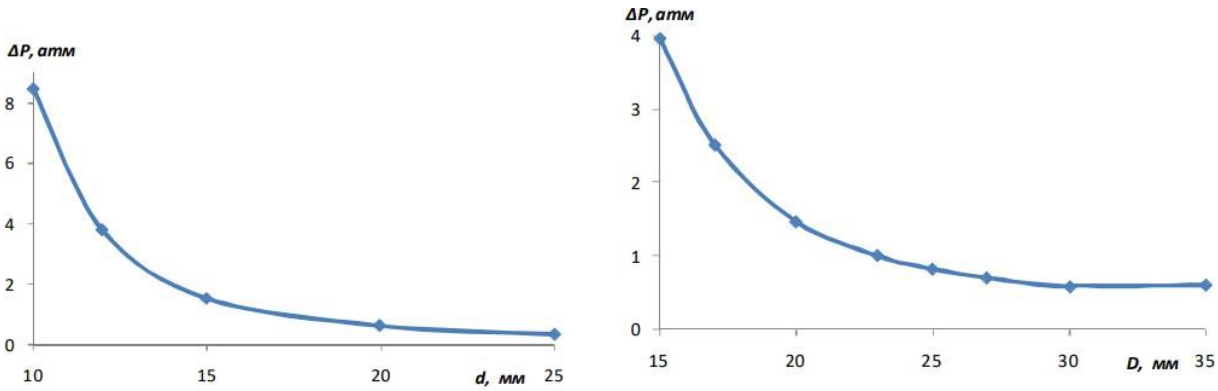


Рисунок 14 – Зависимости перепадов давления для стенки кристаллизатора в узлах P'_0, P'_2, P'_3 от расхода воды



а б

Рисунок 15 – Зависимость перепада давления на кристаллизаторе от диаметра: а – вертикальных каналов; б – горизонтальных каналов

Таким образом, по разработанному в математической модели алгоритму гидравлического расчета стенки кристаллизатора, создана программа, позволяющая производить подсчеты по заданным параметрам

Вывод по третьей главе

Для схемы взаимодействия с последовательным соединением каналов была разработана компьютерная программа Console Interface, написанная на языке программирования C++, основной задачей которой является нахождение потери давления потока, а также скорости движения потока и числа Рейнольдса. Результатами расчета по данному алгоритму являются:

```
m_averageFlowRate = 3.67875
m_reynoldNumber = 140256
    m_flowType = turbulent
m_flowPressureLoss = 262266
```

Листинг 5 – Результаты расчета

При данном расчете выявлена погрешность относительно математической модели, описанной во второй главе.

Для схемы взаимодействия со сложным соединением каналов была использована компьютерная программа “Кристаллизатор-гидравлика”, написанная в MATLAB.

Расход воды, $м^3/ч$	Скорости воды, м/с		
17	5,03	4,95	5,05

Перепады давления в узлах относительно входного давления составили: при подаче воды в стенку кристаллизатора сверху – от 0,0025 до 0,600 атмосфер, при подаче снизу – от 0,0025 до 0,390 атмосфер.

4. Проверка адекватности и достоверности моделей

Рассматривается компьютерная модель расчета гидравлических процессов кристаллизатора МНЛЗ. Произведена оценка адекватности и точности реализованной модели, путем сравнения показателей, полученных на модели, с фактическими данными. За основу реального объекта взяты выходные данные из математической модели Т.П. Лариной, К.Н. Вдовина, И.М. Ячикова.

Оценка адекватности модели по отношению к реальному объекту оценивается по близости результатов расчетов экспериментальным данным. Исходя из этого, принято решение воспользоваться метриками. Метрика – внешний критерий качества, зависящий от предсказанных значений, но не от параметров модели. Метрика качества – это оценка качества нашей модели. Функцией потерь является критерий “обучаемости” нашей модели, т.е. переход к поиску оптимальных параметров модели, возникающих от перехода построении модели к задаче оптимизации.

При сравнении гидравлической работы различных вариантов стенки кристаллизатора со сложным соединением каналов основными характеристиками являются: средний δ и максимальный δ_{max} разбросы скоростей (отдельно взяты в вертикальных каналах).

Под средним разбросом скорости понимается отношение суммы модулей разности скорости в i-ом канале и средней скорости, и произведения количества каналов и средней скорости:

$$\delta = \frac{\sum_{i=1}^{M} |V_i - \bar{V}|}{M\bar{V}}. \quad (12)$$

Под максимальным разбросом скоростей понимается отношение максимального модуля разности скорости в i-ом канале и средней скорости, и средней скорости:

$$\delta_{max} = \frac{max(|V_i - \bar{V}|)}{\bar{V}}, \quad (13)$$

где $\bar{V} = \sum_{i=1}^{M} V_i \,/M$ – средняя скорость в каналах.

Погрешность при подсчете скорости воды в вертикальных каналах

составили $\delta = 0{,}8\ \%$, $\delta_{max} = 1{,}2\ \%$.

Рассмотрим алгоритм гидравлической работы стенки кристаллизатора с последовательным соединением каналов и сравним с полученными расчетными данными из компьютерной программы. Ключевыми характеристиками являются: средняя скорость движения потока и потеря давления потока.

Погрешность при подсчете средней скорости движения потока составила: $\delta = 0{,}001\ \%$. При подсчете потери давления потока: $\delta = 0{,}005\ \%$

Вывод по четвертой главе

Были выполнены расчеты погрешностей по метрикам, результаты которых указали на неточность разработанной компьютерной модели.

Опираясь на то, что все параметры моделирования установлены достоверно, полученный результат по проверке адекватности компьютерной модели можно считать удовлетворительным.

Исходя из этого, можно заключить, что есть возможность доработки данных программ.

ЗАКЛЮЧЕНИЕ

В первой главе в результате анализа предметной области и существующих статей и моделей из зарубежных и отечественных источников был выбран метод реализации поставленной задачи. Если среди иностранных специалистов по большей части рассматривается двумерная и трехмерные модели, то исследователи из Российской Федерации упор ставят на математическую, трехмерную модели, а также на цифровые двойники. В качестве основы была выбрана разработка компьютерной модели на основе математической модели Т.П. Лариной, К.Н. Вдовина, И.М. Ячикова.

Во второй главе был создан алгоритм решения задачи моделирования гидравлических процессов на основе математической модели.

Основной задачей алгоритма при сложном соединении получить зависимости перепада давления, зависящего не только от расхода воды, а также от геометрических характеристик (в нашем случае, диаметры каналов, их длина и шероховатость).

Результатами расчета для потока с последовательным соединением каналов по данному алгоритму являются: скорость движения ω_{Cp} = 3,67 м/с; число Рейнольдса Re = 140256, благодаря этому определяем, что установленный режим является турбулентным, потеря давления потока Δp = 262478,5 Па.

На основе этих данных будет разработана компьютерная модель для расчета скорости движения, числа Рейнольдса, режима течений и потерь давления потока.

В третьей главе для схемы взаимодействия с последовательным соединением каналов была разработана компьютерная программа Console Interface, написанная на языке программирования C++, основной задачей которой является нахождение потери давления потока, а также скорости движения потока и числа Рейнольдса. Результаты расчета по данному алгоритму представлены в таблице 3:

Таблица 3 – Результаты расчета

Параметр	Расчетное значение
Средняя скорость потока ω_{Cp}, м/с	3,67875
Число Рейнольдса	140256
Тип потока	Турбулентный
Потеря давления потока Δp, Па	262266

При данном расчете выявлена погрешность относительно математической модели, описанной во второй главе.

Для схемы взаимодействия со сложным соединением каналов была использована компьютерная программа “Кристаллизатор-гидравлика”, написанная в MATLAB.

Расход воды, м^3/ч	Скорости воды, м/с		
17	5,03	4,95	5,05

Перепады давления в узлах относительно входного давления составили: при подаче воды в стенку кристаллизатора сверху – от 0,0025 до 0,600 атмосфер, при подаче снизу – от 0,0025 до 0,390 атмосфер.

В четвертой главе были выполнены расчеты погрешностей по метрикам, результаты которых указали на неточность разработанной компьютерной модели.

Опираясь на то, что все параметры моделирования установлены достоверно, полученный результат по проверке адекватности компьютерной модели можно считать удовлетворительным.

Исходя из этого, можно заключить, что есть возможность доработки данных программ

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Калягин, Ю.А. Тепловые процессы при непрерывной разливке стали и в оборудовании машин непрерывного литья заготовок: дис. канд. тех. наук: 05.14.04 / Калягин Юрий Александрович; науч.рук. Н.И. Шестаков; ЧГУ. – Череповец, 2005. – 444 с.
2. МНЛЗ, установки непрерывной разливки стали, их модернизация и запчасти к ним // Машпром: [сайт]. – 2024. – URL: https://mashprom.ru/competentions/metall/steel-smelting/_aview_b37/ (дата обращения: 06.05.2024).
3. Машины непрерывного литья заготовок (МНЛЗ) для стали // Урал Индуктор: [сайт]. – 2024. – URL: https://uralinduktor.ru/products/mashiny-nepreryvnogo-litya-zagotovok-mnlz/mnlz-mashiny-nepreryvnogo-litya-zagotovok-dlya-stali/ (дата обращения: 06.05.2024).
4. Типы МНЛЗ и их применение // Инженерные системы: [сайт]. – 2024. – URL: http://engineeringsystems.ru/proektirovanie-metallurgicheskih-zavodov/tipi-mnlz-i-ih-preimushchestva.php (дата обращения: 06.05.2024).
5. Федоткина. М. СОРТАМЕНТ ЧЕРНЫХ МЕТАЛЛОВ / М. Федоткина, Э. Г. Кременчугская, А. П. Якуничкина, Е. И. Морозова. – 2-е изд. – М.: Калужская типография стандартов, 1969. – 328 с.
6. Валуев, Д.В. Непрерывная разливка стали и сплавов: учебное пособие / Н.А.Козырев, Р.А.Гизатулин, Д.В. Валуев; ЮТИ. – Томск: Изд-во ТПУ, 2014. – 406с.
7. Кристаллизатор машины непрерывного литья металла // Findpatent: [сайт]. – 2024. – URL: https://findpatent.ru/patent/205/2058213.html (дата обращения: 06.05.2024).
8. Thomas B. Modeling of the continuous casting of steel – past, present, and future / B. Thomas // Iron and Steel Society. – 2002. – Vol. 33. – P. 795 – 812.
9. Meng Y. Heat – transfer and solidification model of continuous slab casting / Y. Meng, B. Thomas // CON1D. – 2003. – Vol. 34. – P. 685 – 705.

10. Yang J. A Three-Dimensional Real-time Heat Transfer Model for Continuous Casting Blooms / J. Yang, Z. Xie, Z. Hu, H. Meng // ISIJ International. – 2023. – Vol.63(8). – P.1360-1372, DOI:10.2355/isijinternational.ISIJINT-2023-051.
11. Гусев, М.П. Концепт цифрового двойника процесса непрерывного литья слитков / М.П. Гусев, К.Н. Анисимов, С.В. Зарубин, А.М. Лонгинов, И.К. Ужинский // ISCON. – 2018. – С. 223–226.
12. Ларина, Т.П. Расчет гидравлических параметров медных кристаллизаторов со сложным соединением каналов / Т.П. Ларина, К. Н. Вдовин, И.М. Ячиков // Изв. вузов. Цвет. металлургия. – 2015. – №5. – С. 74–79. DOI: dx.doi.org/10.17073/0021-3438-2015-5-74-79.
13. Ячиков, И.М. Моделирование тепловых полей в кристаллизаторе с щелевыми каналами охлаждения / И.М. Ячиков, Н.А. Феоктистов, А.С. Савинов, Т.И. Шафиков, И.В. Михалкина // Теория и технология металлургического производства. – 2022. – № 1 (40). – С. 12–18.
14. Машины непрерывного литья заготовок. Теория и расчет / Под общ. ред. Г.А. Шалаева. – Екатеринбург: Уральский центр ПР и рекламы, 2003. – 320 с.
15. Ячиков, И.М. Сравнение тепловой работы кристаллизаторов МНЛЗ с охлаждающими каналами различной формы / И.М. Ячиков, Т.П. Ларина, К. Н. Вдовин // Изв. вузов. Чер. металлургия. – 2007. – № 11. – С. 55–60.
16. Ячиков, И.М. Гидравлический расчет стенки кристаллизатора / И.М. Ячиков, Т.П. Ларина, К. Н. Вдовин // Свидетельство о государственной регистрации программы для ЭВМ. № 2013618853. – БПБТ. – 2013. – №4. – С. 199.
17. Шафиков, Т.И. Расчет гидравлических параметров водяного охлаждения медной стенки слябового кристаллизатора МНЛЗ при разной схеме подвода и отвода воды / Т.И. Шафиков, И. М. Ячиков, М. Н. Самодурова // ФГАОУ ВО «ЮУрГУ (НИУ)». – Челябинск: Изд-во ЮУрГУ, 2022. – С. 172–180.
18. Курганов, А.М. Гидравлические расчеты систем водоснабжения и водоотведения: Справочник / Под общ. ред. А. М. Курганова. – 3-е изд., перераб. и доп. – Л.: Стройиздат. Ленингр. отд-ние, 1986. – 440 с.: ил.

19. Ячиков, И.М. Непрерывная разливка стали. Расчеты медных кристаллизаторов / И.М. Ячиков, Т.П. Ларина, К. Н. Вдовин. – Магнитогорск: Изд-во Магнитогорск. гос. техн. ун-та им. Г.И. Носова, 2014. – 190 с.
20. Печенегов, Ю.Я. Гидравлические процессы: Примеры расчетов на ЭВМ и задачи для самостоятельного решения: Учебно-методическое пособие к практическим работам по курсу «Процессы и аппараты химической технологии» с применением ЭВМ для студентов специальности 250400 «Химическая технология природных энергоносителей и углеродных материалов» – Саратов: Изд-во Сарат. ун-та, 2009. – 37 с., ил.
21. Вдовин, К.Н. Непрерывная разливка стали: монография / К.Н. Вдовин, И.М. Точилкин, И.М. Ячиков. – СПб.: Лань, 2020. – 732 с.

ПРИЛОЖЕНИЯ

Приложение А. Исходный код программы

Листинг 1 – main.cpp

```
#include <iostream>
#include "ConsoleInterface.h"
#include "MathModel.h"
#include <memory>

int main() {
    setlocale(LC_ALL, "Russian");

    std::unique_ptr<MathModel> mathModel{new MathModel()};

    ConsoleInterface consoleInterface(mathModel->setDataFunc(), mathModel-
>getResultFunc());

    consoleInterface.run();


    std::cin;
    return 0;
}
```

Листинг 2 – MathModel.h

```
#pragma once

#include <functional>
#include <ConsoleInterface.h>
#include <memory>
class MathModel final {
private:
    void calculateVolumeFlowPerSecond();
    void calculateAverageFlowRate();
    void calculateReynoldNumber();
    void calculateLambdaForLaminar();
    void calculateLambdaForTurbulent();
    void calculateSumResistanceCoefficientList();
    void calculateFlowPressureLoss();
public:
    explicit MathModel();
    std::vector<double> getResult();
    void setData(const Params &params);
    std::function<void(const Params &)> *setDataFunc();
    std::function<std::vector<double>()> *getResultFunc();
private:
    std::unique_ptr<std::function<void(const Params &)>> m_setData{nullptr};
    std::unique_ptr<std::function<std::vector<double>()>>
m_getResult{nullptr};
    Params m_params;
    double m_volumeFlowPerSecond{0.0};
    double m_averageFlowRate{0.0};
    double m_reynoldNumber{0.0};
    double m_lambda{0.0};
    double m_sumResistanceCoefficientList{0.0};
    double m_flowPressureLoss{0.0};
};
```

Листинг 3 – MathModel.cpp

```
#include "MathModel.h"

MathModel::MathModel() {
    m_setData.reset(new std::function<void(const Params &)>([&](const Params
&params) {
        setData(params);
    }));

    m_getResult.reset(new std::function<std::vector<double>()>([&]() {
        return getResult();
    }));

}

std::vector<double> MathModel::getResult() {
    return {m_averageFlowRate, m_reynoldNumber, m_flowPressureLoss};
}

void MathModel::setData(const Params &params) {

    m_params = params;

    calculateVolumeFlowPerSecond();
    calculateAverageFlowRate();
    calculateReynoldNumber();
    calculateSumResistanceCoefficientList();
    calculateFlowPressureLoss();

}

std::function<void(const Params &)> *MathModel::setDataFunc() {
    return m_setData.get();
}

std::function<std::vector<double>()> *MathModel::getResultFunc() {
    return m_getResult.get();
}

void MathModel::calculateVolumeFlowPerSecond() {
    m_volumeFlowPerSecond = m_params.consumptionPerHour /
(m_params.compressStress * 3600);
}

void MathModel::calculateAverageFlowRate() {
    m_averageFlowRate = 4 * m_volumeFlowPerSecond / (M_PI *
std::pow(m_params.crystallizerChannelDiameter, 2));

}

void MathModel::calculateReynoldNumber() {
    m_reynoldNumber =
            (m_averageFlowRate * m_params.crystallizerChannelDiameter *
m_params.compressStress) / m_params.viscosity;
    if (m_reynoldNumber < 2320) {
```

Продолжение листинга 3 приложения А

```
        calculateLambdaForLaminar();

} else {
        calculateLambdaForTurbulent();
    }

}

void MathModel::calculateLambdaForLaminar() {
    m_lambda = 64 / m_reynoldNumber;
}

void MathModel::calculateLambdaForTurbulent() {
    m_lambda = 0.316 / std::pow(m_reynoldNumber, -0.25);
}

void MathModel::calculateSumResistanceCoefficientList() {

    for (size_t i = 0; i < m_params.resistanceCoefficientList.size() - 1;
i++) {
        m_sumResistanceCoefficientList +=
m_params.resistanceCoefficientList[i];


    }
    m_sumResistanceCoefficientList += m_params.resistanceCoefficientList[4] *
m_params.pipeCurve;
}

void MathModel::calculateFlowPressureLoss() {
    m_flowPressureLoss = 0.5 * std::pow(m_averageFlowRate, 2) *
m_params.compressStress *
                         (1 + (m_lambda * m_params.crystallizerLength *
m_params.viscosity * 2.98 /
                               m_params.crystallizerChannelDiameter) +
                          m_sumResistanceCoefficientList);
}
```

Окончание листинга 3 приложения А

Листинг 4 – ConsoleInterface.h

```
#pragma once
#include <iostream>
#include <array>
#include <string>
#include <vector>
#include <tuple>
#include <functional>
#include <cstdint>
#include <cmath>
#include <algorithm>

struct Params {
    double consumptionPerHour{12000};
    double compressStress{998};
    double viscosity{0.00089};
    double crystallizerLength{55};
```

```
 double crystallizerChannelDiameter{0.034};

 uint32_t pipeCurve{2};
    std::array<double, 5> resistanceCoefficientList{0.5, 1, 6, 0.5, 1.8};

    static const double epsilon;

    bool operator==(const Params &other) const;
};


enum class Locale : uint8_t {
    Rus = 0x00,
    Eng = 0x01
};


enum class CurrentMenu : uint8_t {
    MainMenu = 0x00,
    EditParams = 0x01,
    SelectEditParams = 0x02,
    ShowParams = 0x03,

    ShowCalculateResult = 0x05,
    SaveResult = 0x06,
    ChangeLanguage = 0x07,
    AlertExit = 0x08,

    consumptionPerHour = 0x09,
    compressStress = 0x0A,
    viscosity = 0x0B,
    crystallizerLength = 0x0C,
    crystallizerChannelDiameter = 0x0D,
    resistanceCoefficientList = 0x0E,
    pipeCurve = 0x0F

};

class ConsoleInterface final {

private:
    using MainMenu = std::vector<std::tuple<std::string, std::wstring>>;
    using ParamsMenu = std::vector<std::tuple<std::string, std::wstring>>;
    using NameParams = std::vector<std::tuple<std::string, std::wstring>>;

    int enterMenuItem();

    void initData();

    void showMainMenu();

    void showEditParamsMenu();

    void showEditSelectParamsMenu();

    void printRusItems(const std::vector<std::tuple<std::string,
```

```
std::wstring>> &mainMenu) const noexcept;

    void printEngItems(const std::vector<std::tuple<std::string,
std::wstring>> &mainMenu) const noexcept;

    void calculate();

    void showParamsMenu(const Params &params);

    void showParams(const Params &params);

    void showResistanceCoefficientList(const Params &params);

    void showCalculateResult();

    void showSaveResult();

    void showChangeLanguage();

    void showAlertExit();

    void showEditConsumptionPerHour();

    void showEditCompressStress();

    void showEditViscosity();

    void showEditCrystallizerLength();

    void showEditCrystallizerChannelDiameter();

    void showEditResistanceCoefficientList();

    void showEditPipeCurve();

    void changeCurrentMenu(CurrentMenu menu);

    void processMenu(int32_t item);

    void processMainMenu(int32_t item);

    void processParamsMenu(int32_t item);

    void processEditParamsMenu(int32_t item);

    void processShowParams(int32_t item);

    void processShowCalculateResult();

    void processSaveResult(uint32_t item);

    void processChangeLanguage(uint32_t item);

    void processAlertExit(uint32_t item);

    void processConsumptionPerHour(uint32_t item);

    void processCompressStress(uint32_t item);
```

Продолжение листинга 4 приложения А

```
void processViscosity(uint32_t item);

    void processCrystallizerLength(uint32_t item);

    void processCrystallizerChannelDiameter(uint32_t item);

    void processResistanceCoefficientList(uint32_t item);

    void processPipeCurve(uint32_t item);

    void saveChangesEditParams();


public:


    explicit ConsoleInterface(std::function<void(const Params &)> *   /*
функция для передачи*/,
                              std::function<std::vector<double>()> *
/*функция для получения результата*/);


    void run();


    ~ConsoleInterface() = default;

    int printMenu();

    void showMenu();


private:
    MainMenu m_mainMenu;
    ParamsMenu m_menuParams;
    NameParams m_nameParams;

    Params m_params;
    Params m_unSaveParams;

    std::function<void(const Params &)> *m_writeParamsFunction;  /* функция
для передачи*/
    std::function<std::vector<double>()> *m_readParamsFunction; /*функция для
получения результата*/

    Locale m_currenLocale{Locale::Eng};
    CurrentMenu m_currentMenu{CurrentMenu::MainMenu};
    bool m_isExit{false};

};
```

Окончание листинга 4 приложения А

Листинг 5 – ConsoleInterface.cpp

```
#include "ConsoleInterface.h"


const double Params::epsilon{0.0001};


void ConsoleInterface::printRusItems(const
std::vector<std::tuple<std::string, std::wstring>> &data) const noexcept {
    for (size_t i = 0; i < data.size(); i++) {
        std::wcout << "[ " << i << " ] - " << std::get<1>(data[i]) <<
std::endl;
    }
};


void ConsoleInterface::printEngItems(const
std::vector<std::tuple<std::string, std::wstring>> &data) const noexcept {
    for (size_t i = 0; i < data.size(); i++) {
        std::cout << "[ " << i << " ] - " << std::get<0>(data[i]) <<
std::endl;
    }
};


ConsoleInterface::ConsoleInterface(std::function<void(const Params &)>
*setParams /* функция для передачи*/,
                                   std::function<std::vector<double>()>
*getResult /*функция для получения результата*/)
        : m_readParamsFunction(getResult),
          m_writeParamsFunction(setParams) {

    //  m_readParamsFunction = getResult;
    initData();

}

bool Params::operator==(const Params &other) const {

    if (resistanceCoefficientList.size() !=
other.resistanceCoefficientList.size()) {
        return false;
    }

    for (size_t i = 0; i < resistanceCoefficientList.size(); i++) {
        if (std::fabs(this->resistanceCoefficientList[i] -
other.resistanceCoefficientList[i]) > epsilon)
            return false;
    }

    return std::fabs(this->consumptionPerHour - other.consumptionPerHour) <=
epsilon &&
           std::fabs(this->compressStress - other.compressStress) <= epsilon
&&
           std::fabs(this->viscosity - other.viscosity) <= epsilon &&
           std::fabs(this->crystallizerLength - other.crystallizerLength) <=
epsilon &&
           std::fabs(this->crystallizerChannelDiameter -
other.crystallizerChannelDiameter) <= epsilon;
}
```

```
void ConsoleInterface::initData() {


    m_nameParams.emplace_back("Consumption per hour (kg/h)", L"Расход за час
(кг/ч)");
    m_nameParams.emplace_back("Density (kg*m^2)", L"Плотность (кг* м^2)");
    m_nameParams.emplace_back("Viscosity (Pa*s)", L"Вязкость (Па*с)"); // +
    m_nameParams.emplace_back("Crystallizer length (m)", L"Длина
кристаллизатора (м)"); // +
    m_nameParams.emplace_back("Crystallizer channel diameter (m)", L"Диаметр
канала кристаллизатора (м)");
    m_nameParams.emplace_back("Count pipe curve", L"Количество колен трубы");
    m_nameParams.emplace_back("Resistance coefficient list", L"Список
коэффициентов сопротивления ");
    m_nameParams.emplace_back("Exit", L"Выход");


    m_mainMenu.emplace_back("Show params", L"Показать параметры");
    m_mainMenu.emplace_back("Edit params menu", L"Меню редактирования
параметров");
    m_mainMenu.emplace_back("Calculate", L"Рассчитать ");
    m_mainMenu.emplace_back("Show result", L"Показать расчеты");
    m_mainMenu.emplace_back("Save results", L"Сохранить результаты");
    m_mainMenu.emplace_back("Change locale", L"Сменить язык");
    m_mainMenu.emplace_back("Exit", L"Выход");


    m_menuParams.emplace_back("Edit", L"Редактировать");
    m_menuParams.emplace_back("Save changes", L"Сохранить изменения");
    m_menuParams.emplace_back("Exit", L"Выход");

}

void ConsoleInterface::run() {

    int32_t selectedItemMenu{0};

    while (!m_isExit) {
        showMenu();

        std::string data;
        std::cin >> data;
        if (m_currentMenu != CurrentMenu::ShowCalculateResult) {
            selectedItemMenu = stoi(data);
        }
        processMenu(selectedItemMenu);
    }

}

void ConsoleInterface::showMenu() {


    switch (m_currentMenu) {
        case CurrentMenu::MainMenu: {
            showMainMenu();
            break;
        }
```

```
case CurrentMenu::EditParams: {
    showEditParamsMenu();
    break;
}
case CurrentMenu::SelectEditParams: {
    showEditSelectParamsMenu();
    break;
}
case CurrentMenu::ShowParams: {
    showParamsMenu(m_params);
    break;
}
case CurrentMenu::ShowCalculateResult: {
    showCalculateResult();
    break;
}
case CurrentMenu::SaveResult: {
    showSaveResult();
    break;
}
case CurrentMenu::ChangeLanguage: {
    showChangeLanguage();
    break;
}
case CurrentMenu::AlertExit: {
    showAlertExit();
    break;
}
case CurrentMenu::consumptionPerHour: {
    showEditConsumptionPerHour();
    break;
}
case CurrentMenu::compressStress: {
    showEditCompressStress();
    break;
}
case CurrentMenu::viscosity: {
    showEditViscosity();
    break;
}
case CurrentMenu::crystallizerLength: {
    showEditCrystallizerLength();
    break;
}
case CurrentMenu::crystallizerChannelDiameter: {
    showEditCrystallizerChannelDiameter();
    break;
}
case CurrentMenu::resistanceCoefficientList: {
    showEditResistanceCoefficientList();
    break;
}
case CurrentMenu::pipeCurve: {
    showEditPipeCurve();
}
default:
    break;
```

```
    }
}
void ConsoleInterface::showMainMenu() {

    switch (m_currenLocale) {

        case Locale::Rus: {
            printRusItems(m_mainMenu);
            break;
        }
        case Locale::Eng: {
            printEngItems(m_mainMenu);
            break;
        }
        default: {
            std::cout << "Locale don't support";
        }
    }
}

void ConsoleInterface::showEditParamsMenu() {
    switch (m_currenLocale) {

        case Locale::Rus: {
            std::wcout << L"Меню редактирования параметров" << std::endl;
            printRusItems(m_menuParams);
            break;
        }
        case Locale::Eng: {
            std::cout << "Menu edit params" << std::endl;
            printEngItems(m_menuParams);
            break;
        }
        default: {
            std::cout << "Locale don't support";
        }
    }
}

void ConsoleInterface::showEditSelectParamsMenu() {
    switch (m_currenLocale) {

        case Locale::Rus: {
            printRusItems(m_nameParams);

            break;
        }
        case Locale::Eng: {
            printEngItems(m_nameParams);
            break;
        }
        default: {
            std::cout << "Locale don't support";
        }

    }
}
```

```
void ConsoleInterface::processMenu(int32_t item) {
    switch (m_currentMenu) {
        case CurrentMenu::MainMenu: {
            processMainMenu(item);
            break;
        }
        case CurrentMenu::EditParams: {
            processParamsMenu(item);
            break;
        }

        case CurrentMenu::SelectEditParams: {
            processEditParamsMenu(item);
            break;
        }
        case CurrentMenu::ShowParams: {
            processShowParams(item);
            break;
        }
        case CurrentMenu::ShowCalculateResult: {
            processShowCalculateResult();
            break;
        }
        case CurrentMenu::SaveResult: {
            processSaveResult(item);
            break;
        }
        case CurrentMenu::ChangeLanguage: {
            processChangeLanguage(item);
            break;
        }
        case CurrentMenu::AlertExit: {
            processAlertExit(item);
            break;
        }
        case CurrentMenu::consumptionPerHour: {
            processConsumptionPerHour(item);
            break;
        }
        case CurrentMenu::compressStress: {
            processCompressStress(item);
            break;
        }
        case CurrentMenu::viscosity: {
            processViscosity(item);
            break;
        }
        case CurrentMenu::crystallizerLength: {
            processCrystallizerLength(item);
            break;
        }
        case CurrentMenu::crystallizerChannelDiameter: {
            processCrystallizerChannelDiameter(item);
            break;
        }
        case CurrentMenu::resistanceCoefficientList: {
```

```
            processResistanceCoefficientList(item);
            break;
        }
        case CurrentMenu::pipeCurve: {
            processPipeCurve(item);
            break;
        }
        default: {
            break;
        }

    }
}

void ConsoleInterface::processMainMenu(int32_t item) {
    switch (item) {

        case 0: {
            changeCurrentMenu(CurrentMenu::ShowParams);
            break;
        }
        case 1: {

            changeCurrentMenu(CurrentMenu::EditParams);
            break;
        }

        case 2: {
            //changeCurrentMenu();
            calculate();
            break;
        }
        case 3: {
            changeCurrentMenu(CurrentMenu::ShowCalculateResult);
            break;
        }
        case 4: {
            changeCurrentMenu(CurrentMenu::SaveResult);
            break;
        }
        case 5: {
            changeCurrentMenu(CurrentMenu::ChangeLanguage);
            break;
        }
        case 6: {
            changeCurrentMenu(CurrentMenu::AlertExit);
            break;
        }
        default: {
            break;
        }


    }

}

void ConsoleInterface::calculate() {
```

```
    (*m_writeParamsFunction)(m_params);

}

void ConsoleInterface::processParamsMenu(int32_t item) {
    switch (item) {

        case 0: {
            //editSelectParamsMenu();
            changeCurrentMenu(CurrentMenu::SelectEditParams);
            break;
        }
        case 1: {
            changeCurrentMenu(CurrentMenu::SaveResult);
            break;
        }
        case 2: {
            changeCurrentMenu(CurrentMenu::MainMenu);
            break;
        }
        default: {
            break;
        }
    }
}
void ConsoleInterface::processEditParamsMenu(int32_t item) {

    switch (item) {
        case 0: {
            changeCurrentMenu(CurrentMenu::consumptionPerHour);
            break;
        }
        case 1: {
            changeCurrentMenu(CurrentMenu::compressStress);
            break;
        }
        case 2: {
            changeCurrentMenu(CurrentMenu::viscosity);
            break;
        }
        case 3: {
            changeCurrentMenu(CurrentMenu::crystallizerLength);
            break;
        }
        case 4: {
            changeCurrentMenu(CurrentMenu::crystallizerChannelDiameter);
            break;
        }
        case 5: {
            changeCurrentMenu(CurrentMenu::pipeCurve);
            break;
        }

        case 6: {
            changeCurrentMenu(CurrentMenu::resistanceCoefficientList);
            break;
        }
```

```
        default: {
            changeCurrentMenu(CurrentMenu::EditParams);
            break;
        }

    }

}

int ConsoleInterface::enterMenuItem() {
    return 0;
}

int ConsoleInterface::printMenu() {
    return 0;
}


void ConsoleInterface::showCalculateResult() {
    auto result = (*m_readParamsFunction)();


    for (auto item: result) {
        std::cout << item << std::endl;
    }


    switch (m_currenLocale) {
        case Locale::Rus: {
            std::wcout << L"\tВведите любую кнопку... " << std::endl;

            break;
        }
        case Locale::Eng: {
            std::cout << "\tPress any button... " << std::endl;
            break;
        }
    }

}

void ConsoleInterface::showSaveResult() {


    switch (m_currenLocale) {
        case Locale::Rus: {
            std::wcout << L"\tТекущие параметры " << std::endl;

            break;
        }
        case Locale::Eng: {
            std::cout << "\tCurrent  Params " << std::endl;
            break;
        }
    }

    showParams(m_params);
```

```
    std::cout << std::endl;
    if (m_params != m_unSaveParams) {
        switch (m_currenLocale) {


            case Locale::Rus: {
                std::wcout << L"\tИзмененные параметры " << std::endl;
                break;
            }
            case Locale::Eng: {
                std::cout << "\tEdited params " << std::endl;
                break;
            }
        }

        showParams(m_unSaveParams);
    }

    switch (m_currenLocale) {
        case Locale::Rus: {
            std::wcout << "[ " << 0 << " ]" << L" Сохранить параметры" <<
std::endl;
            std::wcout << "[ " << 1 << " ]" << L" Выход" << std::endl;
            break;
        }
        case Locale::Eng: {
            std::cout << "[ " << 0 << " ]" << " Save params" << std::endl;
            std::cout << "[ " << 1 << " ]" << " Exit" << std::endl;
            break;
        }
    }

}

void ConsoleInterface::showChangeLanguage() {

    switch (m_currenLocale) {


        case Locale::Rus: {

            std::wcout << "[ " << 0 << " ]" << L" Русский [ текущий ]" <<
std::endl;
            std::wcout << "[ " << 1 << " ]" << L" Английский " << std::endl;
            std::wcout << "[ " << 2 << " ]" << L" Выход" << std::endl;
            break;
        }
        case Locale::Eng: {

            std::cout << "[ " << 0 << " ]" << " Russian " << std::endl;
            std::cout << "[ " << 1 << " ]" << " English [ current ]" <<
std::endl;
            std::cout << "[ " << 2 << " ]" << " Exit" << std::endl;

            break;
        }
```

```
    }


}

void ConsoleInterface::showAlertExit() {

}

void ConsoleInterface::showEditConsumptionPerHour() {

    switch (m_currenLocale) {


        case Locale::Rus: {

            std::wcout << "[ " << 0 << " ]" << std::get<1>(m_nameParams[0])
<< " = "
                       << m_unSaveParams.consumptionPerHour
                       << std::endl;

            std::wcout << "[ " << 1 << " ]" << L" Выход" << std::endl;

            break;
        }
        case Locale::Eng: {
            std::cout << "[ " << 0 << " ]" << std::get<0>(m_nameParams[0]) <<
" = "
                      << m_unSaveParams.consumptionPerHour
                      << std::endl;
            std::cout << "[ " << 1 << " ]" << " Exit" << std::endl;
            break;
        }
    }

}

void ConsoleInterface::showEditCompressStress() {

    switch (m_currenLocale) {


        case Locale::Rus: {

            std::wcout << "[ " << 0 << " ]" << std::get<1>(m_nameParams[1])
<< " = "
                       << m_unSaveParams.compressStress
                       << std::endl;

            std::wcout << "[ " << 1 << " ]" << L" Выход" << std::endl;

            break;
        }
        case Locale::Eng: {
            std::cout << "[ " << 0 << " ]" << std::get<0>(m_nameParams[1]) <<
" = "
                      << m_unSaveParams.compressStress
                      << std::endl;
```

```
            std::cout << "[ " << 1 << " ]" << " Exit" << std::endl;
            break;
        }
    }
}

void ConsoleInterface::showEditViscosity() {

    switch (m_currenLocale) {


        case Locale::Rus: {

            std::wcout << "[ " << 0 << " ]" << std::get<1>(m_nameParams[2])
<< " = "
                       << m_unSaveParams.viscosity
                       << std::endl;

            std::wcout << "[ " << 1 << " ]" << L" Выход" << std::endl;

            break;
        }
        case Locale::Eng: {
            std::cout << "[ " << 0 << " ]" << std::get<0>(m_nameParams[2]) <<
" = "
                      << m_unSaveParams.viscosity
                      << std::endl;
            std::cout << "[ " << 1 << " ]" << " Exit" << std::endl;
            break;
        }
    }
}
void ConsoleInterface::showEditCrystallizerLength() {
    switch (m_currenLocale) {

        case Locale::Rus: {

            std::wcout << "[ " << 0 << " ]" << std::get<1>(m_nameParams[3])
<< " = "
                       << m_unSaveParams.crystallizerLength
                       << std::endl;

            std::wcout << "[ " << 1 << " ]" << L" Выход" << std::endl;

            break;
        }
        case Locale::Eng: {
            std::cout << "[ " << 0 << " ]" << std::get<0>(m_nameParams[3]) <<
" = "
                      << m_unSaveParams.crystallizerLength
                      << std::endl;
            std::cout << "[ " << 1 << " ]" << " Exit" << std::endl;
            break;
        }
    }
}

void ConsoleInterface::showEditCrystallizerChannelDiameter() {
```

```
    switch (m_currenLocale) {


        case Locale::Rus: {

            std::wcout << "[ " << 0 << " ]" << std::get<1>(m_nameParams[4])
<< " = "
                       << m_unSaveParams.crystallizerChannelDiameter
                       << std::endl;

            std::wcout << "[ " << 1 << " ]" << L" Выход" << std::endl;

            break;
        }
        case Locale::Eng: {
            std::cout << "[ " << 0 << " ]" << std::get<0>(m_nameParams[4]) <<
" = "
                      << m_unSaveParams.crystallizerChannelDiameter
                      << std::endl;
            std::cout << "[ " << 1 << " ]" << " Exit" << std::endl;
            break;
        }
    }
}
void ConsoleInterface::showEditResistanceCoefficientList() {

    showResistanceCoefficientList(m_unSaveParams);

    switch (m_currenLocale) {
        case Locale::Rus: {
            std::wcout << "[ " <<
m_unSaveParams.resistanceCoefficientList.size() << " ]" << L"Выход" <<
std::endl;

            break;
        }
        case Locale::Eng: {

            std::cout << "[ " <<
m_unSaveParams.resistanceCoefficientList.size() << " ]" << " Exit" <<
std::endl;
            break;
        }
    }
}

void ConsoleInterface::showResistanceCoefficientList(const Params &params) {
    switch (m_currenLocale) {

        case Locale::Rus: {

            std::wcout << L"\tСписок коэффициентов сопротивления :" <<
std::endl;

            for (size_t i = 0; i < params.resistanceCoefficientList.size();
i++) {
                std::wcout << "[ " << i << " ]" << L"коэффициент
```

```
 сопротивления - " << i << L" = "
                            << params.resistanceCoefficientList[i]
                            << std::endl;
            }


            break;
        }
        case Locale::Eng: {
            std::wcout << "\tResistance coefficient list :" << std::endl;
            for (size_t i = 0; i < params.resistanceCoefficientList.size();
i++) {
                std::wcout << "[ " << i << " ]" << "resistance coefficient -
" << i << " = "
                            << params.resistanceCoefficientList[i]
                            << std::endl;
            }

            break;
        }

    }
}

void ConsoleInterface::showParamsMenu(const Params &params) {
    showParams(params);

    switch (m_currenLocale) {

        case Locale::Rus: {


            std::wcout << "[ " << 0 << " ]" << L" Выход" << std::endl;
            break;
        }
        case Locale::Eng: {

            std::wcout << "[ " << 0 << " ]" << L" Exit" << std::endl;
            break;
        }

    }

}

void ConsoleInterface::showParams(const Params &params) {

    switch (m_currenLocale) {

        case Locale::Rus: {

            std::wcout << std::get<1>(m_nameParams[0]) << L" = " <<
params.consumptionPerHour << std::endl;
            std::wcout << std::get<1>(m_nameParams[1]) << L" = " <<
params.compressStress << std::endl;
            std::wcout << std::get<1>(m_nameParams[2]) << L" = " <<
params.viscosity << std::endl;
            std::wcout << std::get<1>(m_nameParams[3]) << L" = " <<
params.crystallizerLength << std::endl;
```

```
            std::wcout << std::get<1>(m_nameParams[4]) << L" = " <<
params.crystallizerChannelDiameter << std::endl;
            std::wcout << std::get<1>(m_nameParams[5]) << L" = " <<
params.pipeCurve << std::endl;

            std::wcout << L"Список коэффициентов сопротивления :" <<
std::endl;

            for (size_t i = 0; i < params.resistanceCoefficientList.size();
i++) {
                std::wcout << L"\tкоэффициент сопротивления - " << i << L" =
" << params.resistanceCoefficientList[i]
                           << std::endl;
            }


//            std::wcout << "[ " << 0 << " ]" << L" Выход" << std::endl;
            break;
        }
        case Locale::Eng: {
            std::cout << std::get<0>(m_nameParams[0]) << " = " <<
params.consumptionPerHour << std::endl;
            std::cout << std::get<0>(m_nameParams[1]) << " = " <<
params.compressStress << std::endl;
            std::cout << std::get<0>(m_nameParams[2]) << " = " <<
params.viscosity << std::endl;
            std::cout << std::get<0>(m_nameParams[3]) << " = " <<
params.crystallizerLength << std::endl;
            std::cout << std::get<0>(m_nameParams[4]) << " = " <<
params.crystallizerChannelDiameter << std::endl;
            std::cout << std::get<0>(m_nameParams[5]) << " = " <<
params.pipeCurve << std::endl;
            std::wcout << "Resistance coefficient list :" << std::endl;

            for (size_t i = 0; i < params.resistanceCoefficientList.size();
i++) {
                std::wcout << "\tresistance coefficient - " << i << " = " <<
params.resistanceCoefficientList[i]
                           << std::endl;
            }
//            std::wcout << "[ " << 0 << " ]" << L" Exit" << std::endl;
            break;
        }

    }

}

void ConsoleInterface::changeCurrentMenu(CurrentMenu menu) {
    m_currentMenu = menu;
}

void ConsoleInterface::processShowParams(int32_t item) {

    switch (item) {
        case 0: {
            changeCurrentMenu(CurrentMenu::MainMenu);
            break;
```

```
        }
        default: {
            changeCurrentMenu(CurrentMenu::MainMenu);
            break;
        }
    }

}

void ConsoleInterface::processShowCalculateResult() {
    changeCurrentMenu(CurrentMenu::MainMenu);
}

void ConsoleInterface::processSaveResult(uint32_t item) {
    switch (item) {
        case 0: {
            saveChangesEditParams();
            break;
        }
        default: {
            m_currentMenu = CurrentMenu::EditParams;
        }
    }
}

void ConsoleInterface::processChangeLanguage(uint32_t item) {

    switch (item) {
        case 0: {
            m_currenLocale = Locale::Rus;
            break;
        }
        case 1: {
            m_currenLocale = Locale::Eng;
            break;
        }
        default: {
            m_currentMenu = CurrentMenu::MainMenu;
            break;
        }
    }
}

void ConsoleInterface::processAlertExit(uint32_t item) {

}

void ConsoleInterface::processConsumptionPerHour(uint32_t item) {

    switch (item) {
        case 0: {

            std::string data;
            std::cin >> data;
            std::replace(data.begin(), data.end(), '.', ',');
            m_unSaveParams.consumptionPerHour = stod(data);
            changeCurrentMenu(CurrentMenu::consumptionPerHour);
            break;
```

```
        }
        default: {

            changeCurrentMenu(CurrentMenu::SelectEditParams);
            break;
        }

    }

}

void ConsoleInterface::processCompressStress(uint32_t item) {
    switch (item) {
        case 0: {

            std::wstring data;
            std::wcin >> data;
            std::replace(data.begin(), data.end(), '.', ',');
            m_unSaveParams.compressStress = stod(data);
            changeCurrentMenu(CurrentMenu::compressStress);
            break;
        }
        default: {
            changeCurrentMenu(CurrentMenu::SelectEditParams);
            break;
        }

    }

}

void ConsoleInterface::processViscosity(uint32_t item) {
    switch (item) {
        case 0: {

            std::wstring data;
            std::wcin >> data;
            std::replace(data.begin(), data.end(), '.', ',');
            m_unSaveParams.viscosity = stod(data);
            changeCurrentMenu(CurrentMenu::viscosity);
            break;
        }
        default: {
            changeCurrentMenu(CurrentMenu::SelectEditParams);
            break;
        }

    }

}

void ConsoleInterface::processCrystallizerLength(uint32_t item) {
    switch (item) {
        case 0: {

            std::wstring data;
            std::wcin >> data;
```

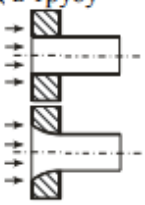
```
            std::replace(data.begin(), data.end(), '.', ',');
            m_unSaveParams.crystallizerLength = stod(data);
            changeCurrentMenu(CurrentMenu::crystallizerLength);
            break;
        }
        default: {
            changeCurrentMenu(CurrentMenu::SelectEditParams);
            break;
        }

    }

}

void ConsoleInterface::processCrystallizerChannelDiameter(uint32_t item) {
    switch (item) {
        case 0: {

            std::wstring data;
            std::wcin >> data;
            std::replace(data.begin(), data.end(), '.', ',');
            m_unSaveParams.crystallizerChannelDiameter = stod(data);
            changeCurrentMenu(CurrentMenu::crystallizerChannelDiameter);
            break;
        }
        default: {
            changeCurrentMenu(CurrentMenu::SelectEditParams);
            break;
        }
    }

}

void ConsoleInterface::processResistanceCoefficientList(uint32_t item) {

    if (item < m_unSaveParams.resistanceCoefficientList.size()) {
        std::wstring data;
        std::wcin >> data;
        std::replace(data.begin(), data.end(), '.', ',');
        m_unSaveParams.resistanceCoefficientList[item] = stod(data);
        changeCurrentMenu(CurrentMenu::resistanceCoefficientList);
    } else {
        changeCurrentMenu(CurrentMenu::SelectEditParams);
    }
}

void ConsoleInterface::saveChangesEditParams() {

    m_params = m_unSaveParams;

}

void ConsoleInterface::processPipeCurve(uint32_t item) {

    switch (item) {
        case 0: {

            std::string data;
```

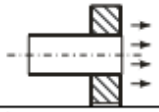
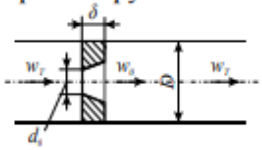
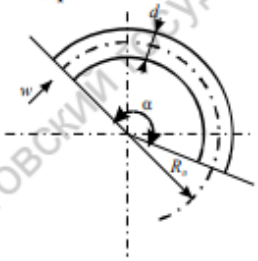
```
            std::cin >> data;
            m_unSaveParams.pipeCurve = stoi(data);
            changeCurrentMenu(CurrentMenu::pipeCurve);
            break;
        }
        default: {
            changeCurrentMenu(CurrentMenu::SelectEditParams);
            break;
        }

    }


}

void ConsoleInterface::showEditPipeCurve() {
    switch (m_currenLocale) {


        case Locale::Rus: {

            std::wcout << "[ " << 0 << " ]" << std::get<1>(m_nameParams[5])
<< " = "
                       << m_unSaveParams.pipeCurve
                       << std::endl;

            std::wcout << "[ " << 1 << " ]" << L" Выход" << std::endl;

            break;
        }
        case Locale::Eng: {
            std::cout << "[ " << 0 << " ]" << std::get<0>(m_nameParams[5]) <<
" = "
                      << m_unSaveParams.pipeCurve
                      << std::endl;
            std::cout << "[ " << 1 << " ]" << " Exit" << std::endl;
            break;
        }
    }
}
```

Приложение Б. Табличные значения

Таблица Б1 – Коэффициенты местных сопротивлений

Коэффициенты местных сопротивлений

<table>
<tr><th>Вид сопротивления</th><th>Значение коэффициента местного сопротивления ξ</th></tr>
<tr><td>Вход в трубу
  </td><td>С острыми краями ξ = 0,5

С закруглёнными краями ξ = 0,2</td></tr>
<tr><td>Выход из трубы
  </td><td>ξ=I</td></tr>
<tr><td>Диафрагма с острыми краями в прямой трубе
 
d_0 – диаметр отверстия диафрагмы; δ – толщина диафрагмы; ω_0 – средняя скорость потока в отверстии; ω_T – средняя скорость в трубе; D – диаметр трубы.</td><td>При $\frac{\delta}{d_o} = 0 \div 0{,}015$ потеря давления
$$\Delta p = \xi \cdot \frac{\rho \cdot \omega_T^2}{2}.$$
Значение ξ определяется по таблице
<table>
<tr><td>m</td><td>0,02</td><td>0,04</td><td>0,06</td><td>0,08</td><td>0,10</td><td>0,12</td><td>0,14</td><td>0,16</td></tr>
<tr><td>ξ</td><td>7000</td><td>1670</td><td>730</td><td>400</td><td>245</td><td>165</td><td>177</td><td>86,0</td></tr>
<tr><td>m</td><td>0,48</td><td>0,20</td><td>0,22</td><td>0,34</td><td>0,26</td><td>0,28</td><td>0,30</td><td>0,34</td></tr>
<tr><td>ξ</td><td>65,0</td><td>51,5</td><td>40,0</td><td>32,0</td><td>26,8</td><td>22,3</td><td>18,2</td><td>13,1</td></tr>
<tr><td>m</td><td>0,4</td><td>0,5</td><td>0,6</td><td>0,7</td><td>0,8</td><td>0,9</td><td></td><td></td></tr>
<tr><td>ξ</td><td>8,25</td><td>4,00</td><td>2,00</td><td>0,97</td><td>0,42</td><td>0,13</td><td></td><td></td></tr>
</table>
$$m = \left(\frac{d_0}{D}\right)^2$$</td></tr>
<tr><td>Отвод круглого или квадратного сечения
 
d – внутренний диаметр трубопровода; R_0 – радиус изгиба трубы.</td><td>Коэффициент сопротивления ξ =A·B определяется по таблицам:
<table>
<tr><td>Угол α, градусы</td><td>20</td><td>30</td><td>45</td><td>60</td><td>90</td><td>110</td><td>130</td><td>150</td><td>180</td></tr>
<tr><td>A</td><td>0,31</td><td>0,45</td><td>0,60</td><td>0,781</td><td>1,0</td><td>1,13</td><td>1,2</td><td>1,28</td><td>1,4</td></tr>
</table>
<table>
<tr><td>R_0/d</td><td>1,0</td><td>2,0</td><td>4,0</td><td>6,0</td><td>15</td><td>40</td><td>50</td></tr>
<tr><td>B</td><td>0,21</td><td>0,15</td><td>0,11</td><td>0,09</td><td>0,06</td><td>0,04</td><td>0,03</td></tr>
</table></td></tr>
<tr><td>Колено 90^0, стандартное чугунное</td><td>
<table>
<tr><td>Условный проход, мм</td><td>12,5</td><td>25</td><td>37</td><td>50</td></tr>
<tr><td>ξ</td><td>2,2</td><td>2,0</td><td>1,6</td><td>1,1</td></tr>
</table></td></tr>
<tr><td>Вентиль</td><td>Значения ξ при полном открытии вентиля:</td></tr>
</table>

Продолжение таблицы Б.1 приложения Б

нормальный

D, мм	13	20	40	80	100	150	200	250
ξ	10,8	8,0	4,9	4,0	4,1	4,4	4,7	5,1

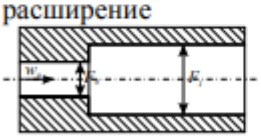
Кран пробочный

Условный проход, мм	13	19	25	32	38	50 и выше
ξ	4	2	2	2	2	2

Задвижка

Условный проход, мм	15-100	175-200	300 и выше
ξ	0,5	0,25	0,14

Внезапное расширение



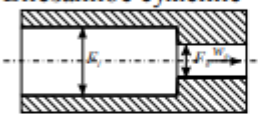
F_0 – площадь меньшего поперечного сечения; F_1 – площадь большего поперечного сечения; ω_0 – скорость потока в меньшем сечении диаметром d_0.

Значения ξ

$\mathrm{Re} = \frac{\omega_o \cdot d_o}{\nu}$	F_0/F_1					
	0,1	0,2	0,3	0,4	0,5	0,6
10	3,1	3,1	3,1	3,1	3,1	3,1
100	1,70	1,40	1,20	1,10	0,90	0,80
1000	2,0	1,60	1,30	1,05	0,90	0,60
3000	1,0	1,0	0,70	0,60	0,40	0,20
3500 и более	0,81	0,64	0,50	0,36	0,25	0,16

$$\Delta p_{рас} = \xi \cdot \frac{\rho \cdot \omega_0^2}{2}$$

Внезапное сужение



F_0 – площадь меньшего поперечного сечения; F_1 – площадь большего поперечного сечения; ω_0 – скорость потока в меньшем сечении диаметром d_0.

Значения ξ

$\mathrm{Re} = \frac{\omega_o \cdot d_o}{\nu}$	F_0/F_1					
	0,1	0,2	0,3	0,4	0,5	0,6
10	5,0	5,0	5,0	5,0	5,0	5,0
100	1,30	1,20	1,10	1,00	0,90	0,80
1 000	0,64	0,50	0,44	0,35	0,30	0,21
10 000	0,50	0,40	0,35	0,30	0,25	0,20
100 000	0,45	0,40	0,35	0,30	0,25	0,20

$$\Delta p_{сух} = \xi \cdot \frac{\rho \cdot \omega_0^2}{2}$$

Окончание таблицы Б.1 приложения Б