Билет 4.

1. Способы решения, полученных в результате применения метода конечных разностей, систем алгебраических уравнений.

Граничные задачи.

При постановке задачи этого типа должны быть даны ответы на следующие вопросы:
Каковы переменные?
Каким уравнениям удовлетворяет решение внутри интересующей нас области?
Каким уравнениям удовлетворяет решение в точках на границе? (Здесь также могут быть условия Дирихле или Неймана, но могут быть и гораздо более сложные условия.)
В отличие от задач с начальными условиями стабильность алгоритма достигается достаточно легко. Самое важное требование к алгоритму, используемому для решения этих задач –*эффективность.*
Все условия в граничных задачах должны быть выполнены «одновременно», поэтому решение таких задач сводится к решению системы большого количества алгебраических уравнений. Если эти уравнения оказываются нелинейными, они линеаризуются или решаются методом итераций, т.е. не теряя общности можно считать, что задача сводится к решению специальной системы, содержащей большое количество линейных уравнений.
Рассмотрим решение уравнения (3) конечно-разностным методом. Мы представляем функцию u(x,y) значениями в узлах прямоугольной сетки

$$x_j = x_0 + j\Delta \quad j=0,1,\ldots,J$$
$$y_l = y_0 + l\Delta \quad l=0,1,\ldots,L \qquad (1.4)$$

где Δ- шаг сетки (одинаковый и по x и по y). Вторые производные можно заменить следующими выражениями

$$\frac{u_{j+1,l}-2u_{j,l}+u_{j-1,l}}{\Delta^2}+\frac{u_{j,l+1}-2u_{j,l}+u_{j,l-1}}{\Delta^2}=\rho_{j,l} \qquad (1.5)$$

или

$$u_{j+1,l}+u_{j-1,l}+u_{j,l+1}+u_{j,l-1}-4u_{j,l}=\Delta^2\rho_{j,l} \qquad (1.6)$$

Чтобы записать систему линейных уравнений в матричной форме перенумеруем точки двумерной сетки одномерной последовательностью

$I = j*(L+1)+l$ для $j=0,1,\ldots,J$, $l=0,1,\ldots,L$ (1.7)
Уравнение (6) принимает вид

$$u_{i+L+1}+u_{i-(L+1)}+u_{i+1}+u_{i-1}-4u_i=\Delta^2\rho_i \qquad (1.8)$$

оно справедливо для внутренних точек области j=1,2,…,J-1; l=1,2,…L-1
Граничные точки области, в которых заданы функции или производные, приведены ниже

j=0 [i.e., i =0 ,…,L]
j=J [i.e., i =J (L +1),…,J (L +1)+L] (1.9)
l=0 [i.e., i =0 ,L +1 ,…,J (L +1)]
l=L [i.e., i =L,L +1+L,…,J (L +1)+L]

всю эту информацию (о значениях на границе) перенесем в правую часть уравнения (8) и уравнение примет вид:

$$\mathbf{A*u=b} \qquad (1.10)$$

Общее эллиптическое уравнение второго порядка при построении разностной схемы приводит к матрице аналогичной описанной выше.
Существуют три различных подхода к решению уравнения (1.10): Методы релаксации, «быстрые методы» (методы Фурье) и прямые матричные методы.

Методы релаксации явно используют структуру разреженной матрицы **A.** Матрица делится на две части

$$\mathbf{A=E-F} \qquad (1.11)$$

Где E легко инвертируется, а F – то что осталось, тогда:

$$\mathbf{E*u=F*u+b} \qquad (1.12)$$

При использовании метода релаксации сначала выбирается начальное приближение, а затем итеративно вычисляется следующее

$$E \cdot u^{(r)} = F \cdot u^{(r-1)} + b \qquad (1.13)$$

E легко инвертируемо, так что итерации быстрые, Метод релаксации обсудим позднее.
Так называемые быстрые методы [ ] применимы только для специального класса уравнений: с постоянными коэффициентами, или, в более общем случае к уравнениям с разделяющимися переменными, Кроме того, границы области должны совпадать с координатными линиями. Заметим однако, что существуют многосеточные методы релаксации, которые могут быть быстрее «быстрых» методов.
При помощи матричных методов прямо решают уравнение

$$\mathbf{A \cdot x = b} \qquad (1.14)$$

Конкретное решение очень сильно зависит от вида матрицы. При решении надо учитывать разреженность матрицы, в противном случае задача осложняется большими размерами матрицы, так при сетке 100*100 надо найти 10000 $u_{j,l}$, что приводит к матрице **A** размером 10000*10000. Если A симметрична и положительно определена, как, например в случае эллиптического уравнения, то можно использовать алгоритм сопряженного градиента [ ].
Кроме метода конечных разностей существует множество других методов решения уравнений с частными производными: метод конечных элементов, Монте-Карло, вариационные методы и другие

2. Диаграммы кооперации для системных операций.

В состав языка UML входят диаграммы взаимодействий (interaction diagrams). Они иллюстрируют способ взаимодействия объектов с помощью сообщений и обеспечивают выполнение необходимых задач. Диаграммы взаимодействий создаются на стадии проектирования цикла разработки. Их создание зависит от следующих созданных ранее артефактов.

- Концептуальная модель. С ее помощью разработчик может определить программные классы, соответствующие понятиям. Объекты этих классов участвуют во взаимодействиях, иллюстрируемых на диаграммах взаимодействий.

• Описание системных операций. С помощью описаний разработчик идентифицирует обязанности и постусловия, которым должны удовлетворять диаграммы взаимодействий.

• Реальные (или идеальные) прецеденты. Из описания прецедентов разработчик может почерпнуть информацию о том, выполнению каких задач должны удовлетворять диаграммы взаимодействий. Эта информация дополняет данные, содержащиеся в описаниях системных операций.

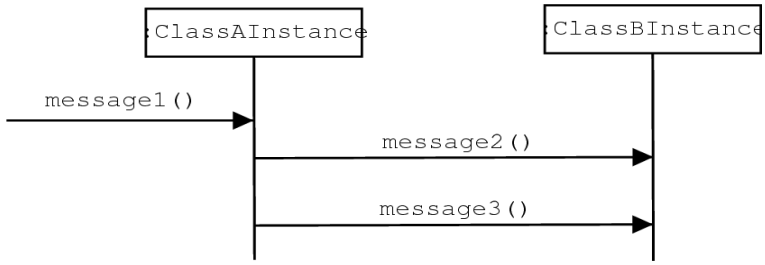
Диаграмма взаимодействий иллюстрирует процесс обмена сообщениями между экземплярами (и классами) в модели классов. Отправной точкой такого взаимодействия является удовлетворение постусловий в описаниях операций.

В языке UML определено два типа диаграмм взаимодействий, которые могут использоваться для иллюстрации либо похожего, либо идентичного обмена сообщениями.

1. Диаграммы кооперации (collaboration diagram).
2. Диаграммы последовательностей (sequence diagram).

Диаграммы кооперации иллюстрируют взаимодействие объектов в формате графа или сети.

Диаграммы последовательностей иллюстрируют взаимодействие в форме, показанной ниже.



Диаграммы кооперации являются исключительно выразительными и способны представлять больше информации, вытекающей из контекста. Кроме того, диаграммы кооперации с точки зрения занимаемого пространства оказываются более экономными. Однако обе системы обозначений позволяют представить подобные конструкции.

Диаграммы взаимодействий — это важный артефакт

Для успешного конструирования диаграмм взаимодействий принципы разработки предварительно можно систематизировать и проанализировать. Такой подход к пониманию и использованию этих принципов основывается на шаблонах (patterns), представляющих собой структурированные рекомендации и принципы.

Создание диаграмм кооперации. При создании диаграмм кооперации руководствуйтесь следующими рекомендациями.

1. В текущем цикле разработки создавайте отдельную диаграмму для каждой системной операции. Для сообщения системной операции разрабатывайте диаграмму таким образом, чтобы это сообщение было входным.

2. Если диаграмма оказалась слишком сложной (например, ее очень трудно разместить на листе бумаги формата А4), разбейте ее на диаграммы меньшего размера.

3. В качестве отправной точки используйте обязанности и постусловия, указанные в описании операции, а также описание прецедентов. Разрабатывайте диаграмму взаимодействий с учетом решения этих задач. Для создания профессиональных диаграмм применяйте GRASP и другие шаблоны,

В прецедентах неявно представлены системные события, которые явно отображаются на диаграммах последовательностей

• Наилучшие исходные предположения о результатах выполнения системных операций описываются в контрактах

• Системные операции определяют сообщения, которые являются отправной точкой создания диаграмм взаимодействий; эти диаграммы иллюстрируют, как объекты взаимодействуют между собой и обеспечивают выполнение поставленных задач

В языке UML для иллюстрации экземпляров объектов используется простой и непротиворечивый подход.

• Для экземпляра любого элемента языка UML (класса, исполнителя и т.д.) используется то же графическое обозначение, что и для типа, однако при этом соответствующая определяющая строка подчеркивается.

Таким образом, для отображения экземпляра класса на диаграмме взаимодействий используется обычное, графическое, условное обозначение класса, однако при этом его имя подчеркивается. Кроме того, на диаграмме кооперации перед именем класса всегда должно указываться двоеточие (:).

Связь (link) является соединением между двумя экземплярами классов, определяющим некоторую форму перемещения и видимости между ними. Более строго можно сказать, что связь является экземпляром ассоциации. При взаимодействии клиента с сервером имеется два таких экземпляра. Существование маршрута перемещения от клиента к серверу означает, что сообщения могут передаваться от клиента серверу.

Передаваемые между объектами сообщения представляются в виде помеченных стрелок над линиями связей. Над одной линией связи может быть указано любое количество сообщений. Для отображения порядка следования сообщений в текущем потоке управления рядом с сообщением приводится порядковый номер.

Параметры сообщения могут указываться внутри круглых скобок, находящихся за именем сообщения Дополнительно может быть указан также тип параметра.

Для сообщения может быть указано возвращаемое значение. Для этого перед сообщением нужно добавить имя переменной с этим значением, а также символ операции присваивания. Дополнительно может быть указан также тип возвращаемого значения.

В языке UML определен стандартный синтаксис описания сообщений. return:= message(parameter: parameterType) : returnType. Сообщение может передаваться объектом самому себе.

Итерационный процесс можно отобразить, указав за порядковым номером сообщения символ "*". Это означает, что сообщение передается получателю несколько раз, в цикле . Можно также указать циклическое условие, определив тем самым, что значения возвращаются многократно. Для представления нескольких сообщений, передаваемых в одном и том же цикле (например, набора сообщений в цикле for), повторяйте условие итерации для каждого сообщения.

Сообщением создания объекта, не зависящим от используемого языка, является create (создать). Оно передается создаваемому экземпляру объекта. Дополнительно новый экземпляр объекта может включать символ "new". Сообщение new может содержать параметры, определяющие передаваемые начальные значения.

Порядок передачи сообщений иллюстрируется с помощью порядковых номеров (sequence number). При этом используется следующая схема нумерации.

1. Первое сообщение не нумеруется. Таким образом, сообщение msg1( ) является непронумерованным.
2. Порядок и вложенность последующих сообщений отображается в соответствии с принятой схемой нумерации. При ее использовании к вложенным сообщениям добавляется номер. Вложенность означает, что к номеру исходящего сообщения добавляется номер входящего сообщения.
Условное сообщение изображается с помощью номера, за которым в квадратных скобках указывается условное выражение, аналогичное условию цикла. Сообщение передается только в том случае, если условный оператор возвращает значение true.
Представление коллекций. Сложный объект обычно реализуется в виде группы экземпляров объектов, содержащихся в контейнере или объекте-коллекции, например в объекте vector стандартной библиотеки шаблонов (STL — Standard Template Library) языка C++, объекте Vector языка Java. Однако такие объекты могут и отсутствовать. В этом случае сложный объект будет представлять логический набор экземпляров объектов.
Сообщение, передаваемое сложному объекту, направляется самому объекту. В языке UML версии 1.1 сообщения, передаваемые сложному объекту, не направляются каждому его элементу (как было в предыдущих версиях языка).
Сообщения могут передаваться самому классу, а не его экземплярам. Это может понадобиться, например, для вызова методов класса. Например, в языке Java такие методы являются статическими, а в языке Smalltalk — методами класса.
Все достаточно просто. Сообщение изображается как обычно, однако в условном обозначении класса его имя не подчеркнуто. Тем самым указывается, что это сообщение передается самому классу, а не его экземпляру. Очень важно, чтобы там, где это нужно, имена экземпляров были подчеркнуты. В противном случае передаваемые сообщения могут быть неверно интерпретированы.