**Иерархическое управление большими системами.**

Большая система, как это кратко было описано в главе 1, - это сложная система, составленная из множества компонентов или меньших подсистем, которые выполняют свои функции, имеют общие ресурсы, и управляемая взаимосвязанными целями и ограничениями (Machmoud, 1977; Jamshidi, 1983). Хотя взаимодействие подсистем может быть организованно в различных формах, одна из общеизвестных – это иерархическая, которая естественна для экономики, менеджмента, в управлении предприятиями, в смешанных отраслях промышленности, таких как роботостроение, производство нефти, стали и бумаги. В этих иерархических структурах, подсистемы расположены на уровнях с различными степенями иерархичности. Подсистема на каком-либо уровне управляет или координирует подсистемы, расположенные на уровне ниже ее, и, в свою очередь, управляется или координируется подсистемой расположенной уровнем выше. Рисунок 4.1 показывает типичную иерархическую (многоуровневую) систему. Верхний уровень управления, иногда его называют координатор высшего уровня (supremal coordinator), можно сравнить с советом директоров корпорации, в то время как другие уровни можно сравнить с президентом, вице-президентом, директорами и т.д. Низший уровень может быть, например, управляющим завода, директором магазина и т.д. тогда как сама большая система – это корпорация. Несмотря на то, что представление иерархической структуры кажется вполне естественным, ее точное поведение еще не совсем изучено, из за того, что сделано мало исследований в области больших систем (March and Simon, 1958). Mesarovic и др. (1970) представили один из самых ранних формальных количественных подходов к иерархической (многоуровневой) системе.С тех пор было сделан много работ в этой области (Schoeffler and Lasdon, 1966; Benveniste et al., 1976; Smith and Sage, 1973; Geoffrion, 1970; Schoeffler, 1971; Pearson, 1971; Cohen and Jolland, 1976; Sandell et al., 1978; Singh,1980; Jamshidi, 1983; Huang and Shao, 1994a,b). Заинтересованный читатель может найти относительно исчерпывающую информацию об управлении многоуровневыми системами и их применении в работе Mahmoud (1977).

В этом разделе дано описание понятия «иерархия», свойств и типов иерархических процессов и представлены некоторые причины для их существования. Полная оценка иерархических методов представлена в разделе 4.6.

Ниже представлены основные свойства иерархических систем, хотя они не общеприняты:

1. Иерархическая система состоит из управляющих блоков, которые организованны по принципу пирамиды.
2. У системы есть общая цель, которая может совпадать или не совпадать с целью отдельных компонентов системы.
3. Различные уровни иерархии системы многократно обмениваются информацией между собой (обычно вертикально).
4. С увеличением уровня временной диапазон тоже увеличивается, то есть компоненты нижних уровней быстрее, чем компоненты верхних.

В иерархических (многоуровневых системах) можно выделить три основные структуры, в зависимости от параметров модели, искомых переменных, поведения и окружающей среды, изменчивости и существования множества взаимоисключающих целей и задач.

1. Многопластовая иерархическая структура. В этой многоуровневой структуре уровни называют пластами. Подсистемы нижнего уровня дают более точное описание большой системы, чем подсистемы верхнего уровня.

2. Многослойная иерархическая структура. Эта структура является результатом сложности процесса регулирования. Задачи управления распределены вертикально, как показано на рисунке 4.2 (Singh and Titli, 1978). В многослойной системе, которая изображена на рисунке, регуляция (на первом уровне) является прямым управлением, а за ним следует оптимизация (вычисление контрольных точек регуляторов), адаптация (непосредственная адаптация закона управления и модели управления) и самоорганизация (выбор модели и управление как функция параметров окружающей среды).

3. Многозвенная иерархическая система. Это самая распространенная из всех трех структур; она состоит из нескольких подсистем, которые располагаются на уровнях таким образом, что каждый уровень (как описано выше) может управлять подсистемами нижнего уровня, и управляется подсистемами верхних уровней. Эта структура, изображенная на рис 4.1, принимает во внимание взаимоисключающие цели и задачи различных подуровней. Другими словами, ступени высшего уровня достигают взаимоисключающих целей путем ослабления взаимодействия между ступенями низшего уровня. Распределение задачи управления данной структуры показано на рисунке 4.2 и, в отличие от многослойной структуры, – горизонтально.

Кроме вертикального и горизонтального распределения задач управления, существует третий способ – временное или функциональное распределение. Это распределение, дающее подсистемам функциональную оптимизацию проблемы, заключается в декомпозиции задачи на конечное число простых задач оптимизации на нижнем уровне и в результате дает немалое сокращение вычислений. Эта схема использовалась для иерархического управления дискретными системами у Jamshidi (1983).

Далее в этой главе говорится о том, как можно эффективно управлять иерархическими системами, используя процессы, известные как декомпозиция и согласование. Эти два процесса представлены на рис 4.3. В итоге, определение иерархического управления: (а) декомпозиция – разделение системы на множество подсистем, и (б) согласование работы этих подсистем, пока не будет достигнуто оптимальное управление всей системой (посредством многоуровневого итеративного алгоритма).

В разделе 4.2 описана возможность применения согласования для иерархических систем. раздел 4.3 посвящен управлению по разомкнутому контуру. Управлению по замкнутому контуру посвящен раздел 4.4, так же в нем даны определения «interaction prediction» и метода структурных возмущений. В разделе 4.5 описано иерархическое управление, основанное на разложение на ряды Тейлора и Чебышева. Проблема управления решается линейными алгебраическими уравнениями. На примерах показаны различные методы решений. Оптимизация линейных и нелинейных иерархических систем описана в главе 6. раздел 4.6 содержит дальнейшее развитие методов иерархического управления.

**4.2. Согласование иерархических структур.**

Как было сказано в предыдущем параграфе, большие системы могут быть иерархически управляемы, для чего сначала надо провести декомпозицию на подсистемы и, затем согласовать полученные подзадачи, преобразовывая сложную систему в многоуровневую. Этого преобразования можно достичь различными путями. Однако, все эти пути, по сути, есть комбинация всего двух отдельных подходов: согласование модели (feasible) и согласование цели (dual-feasible). и методы описаны в следующих двух параграфах, на примере статической оптимизации системы, состоящей из двух подсистем (динамическое программирование).

**4.2.1 Метод согласования модели.**

Рассмотрим следующую статическую оптимизационную задачу:

 (4.2.1)

 (4.2.2)

где x – вектор состояния системы, u – вектор управления, y – вектор взаимодействия между подсистемами. Декомпозируем задачу и ее целевую функцию на две подсистемы:

 (4.2.3)

и

 (4.2.4)

где xi, ui, yi – управляющие векторы системы и выходные векторы i-й подсистемы, соответственно. Такая декомпозиция дает функцию производительности (функционал) для каждой подсистемы. Однако, вектора yi, i=1,2 подсистем взаимосвязаны. Цель метода согласования модели – преобразовать общую задачу в двухуровневую задачу установки значений векторов y1 и y2 в некоторые значения wi, i=1,2:

 (4.2.5)

Данная задача разделяется на две последовательные подзадачи:

Первый уровень подсистемы i:

 (4.2.6)

 (4.2.7)

Второй уровень:

 (4.2.8)

Эту минимизацию можно представить как:

 (4.2.9)

 (4.2.10)

В этой процедуре согласования переменная wi, которая фиксирует изменения переменной yi, называется переменной согласования. Кроме того, внутренние изменения фиксируются добавлением вынужденной составляющей мат модели, эта процедура называется согласованием модели. Другими словами, сам факт представления всех промежуточных значений переменных x, u и y, так же называется метод точной декомпозиции. Следовательно, система может оперировать с теми промежуточными переменными, которые ведут к локальной оптимизации. Первый уровень задачи фиксируется точным взаимодействием переменных с первоначальной задачей оптимизации, пока определяется задача выделения согласующих переменных второго уровня.

* + 1. **Метод согласования цели.**

Рассмотрим задачу статической оптимизации (4.2.1)-(4.2.2). В методе согласования цели удаляются все связи между подсистемами. Выходную переменную i-й подсистемы обозначим как yi, а входную – zi. Пусть все связи между подсистемами отсутствуют, т.е. . При этом условии, zi действует как случайно управляемая переменная и оптимизирует подсистему подобно x, u и y. Кроме того, задача оптимизации, рассмотренная в предыдущем параграфе, решена для уже разделенной на две подсистемы системой, где разделены взаимодействия подсистем и их целевые функции. Далее необходимо убедится, что все подсистемы вместе решают первоначальную задачу, для этого должно выполнятся правило уравновешенного взаимодействия, т.е. независимого выбора yi и zi для решения (Mesarovic и др., 1969; Schoeffler, 1971).

Опишем процедуру декомпозиции задачи на отдельные подзадачи, которые содержат задачи первого уровня. Второй уровень решения управляет первым, опираясь на правило уравновешенного взаимодействия. С точки зрения математики, это многоуровневую формулировку можно записать с помощь параметра веса , который определяет штраф системы, где не сбалансировано взаимодействие. Целевая функция примет вид:

 (4.2.11)

где – вектор параметров веса (положительных и отрицательных), которые изменяют целевую функцию в зависимости от разности y-z. Введем переменную z, тогда решение системы примет вид:

 (4.2.12)

 (4.2.13)

Набор допустимых системных переменных определяется так:

 (4.2.14)

Целевая функция минимизируется посредством S0:

 (4.2.15)

Приняв за штраф и учитывая (4.2.11)-(4.2.13) , задача первого уровня формулируется как:

Подсистема 1:

 (4.2.16)

 (4.2.17)

Подсистема 2:

 (4.2.18)

 (4.2.19)

Второй уровень управляет согласованием переменной , исходя из невязки по выходу:

 (4.2.20)

Из задачи второго уровня ясно, что согласующей переменной х управляют до тех пор, пока ошибка е не достигнет нуля, т.е. баланс взаимодействия поддерживается посредством целевой функции задач первого уровня (4.2.16) и (4.2.18) и через переменную , отсюда и название – согласование цели. На рис 4.4 изображено двухуровневое решение через согласование цели. Читатель может сравнить схемы 4.4 и 4.5.

Позже мы увидим, что переменную согласования а можно истолковать как вектор управления Лагранжа и задачу второго уровня можно решить через хорошо известные итеративные поисковые алгоритмы, такие как метод градиента, Ньютона и скоростного градиента.

**4.3 Иерархическое управление линейными системами.**

В этом разделе формулировка согласования цели для многоуровневых систем применяется к большим линейным непрерывным системам в контексте управления по разомкнутому циклу. Кроме подхода с балансом взаимодействия обсуждается так же другая схема, известная как метод наблюдения взаимодействия.

Пусть большая динамическая взаимосвязанная система представлена в виде следующего уравнения состояния:

 (4.3.1)

где х и u – это векторы состояния (n x l) и управления (m x l). Принято считать, что система может быть разложена на N взаимосвязанных подсистем si, i=1,…,N, и управление состояния i-й подсистемы может быть представлено как:

, (4.3.2)

где x, u, xi, ui – имеют размерность n, m, ni, mi, соответственно, а gi – представляет взаимосвязи в i-й подсистеме, и:

 (4.3.3)

 (4.3.4)

Задачей оптимального управления является поиск управляющих векторов u1,…,uN, таких, что оценочная функция

 (4.3.5)

минимизирует объект (4.3.1) и подходящая область:

 (4.3.6)

Учитывая возможность декомпозиции системы (4.3.1) на N соединенных подсистем (4.3.2), можно разложить ценовую функцию (4.3.6) и взаимосвязи gi(x,t) (4.3.2), как:

 (4.3.7)

 (4.3.8)

 (4.3.9)

где zi – вектор содержащий линейную (или нелинейную) комбинацию состояний N подсистем. Исходя из описанных предположений, задача оптимального управления большой системой может быть записана как:

 (4.3.10)

 (4.3.11)

 (4.3.12)

 (4.3.13)

Эта проблема, известная как иерархическое управление, была решена двухуровневой оптимизацией статистической задачи в предыдущем параграфе. Применение двухуровневого согласования цели для больших линейных систем описано далее.

**4.3.1 Двухуровневое согласование линейных систем**

Рассмотрим большую линейную стационарную систему:

 (4.3.14)

Система может быть декомпозирована как:

 (4.3.15)

где вектор взаимодействия (k x l), записанный как:

 (4.3.16)

это линейная комбинация состояний N-1 подсистем, и Gij – это матрица ni x nj. Первоначальная задача оптимального управления системой сводится к оптимизации N подсистем, которые удовлетворяют (4.3.15)-(4.3.16) и минимизируют:

 (4.3.17)

где Qi – это неотрицательно определенная матрица ni x ni, Ri и Vi – это положительно определенные матрицы mi x mi и ki x ki , где

 (4.3.18)

Физическая интерпретация последнего слагаемого в интеграле (4.3.17) – это неточность в данной точке. Фактически, определяя это слагаемое, как будет видно дальше, мы избегаем выраженных управлений. «Согласование цели» и «баланс взаимодействия» использованные у Mesarvic и др. (1970), так же известны как задача «linear-quadratic» у Pearson (1971) и передача у Singh (1980) и Jamshidi (1983).

В этой декомпозиции большой взаимосвязанной линейной системы общие коэффициенты связи между ее N подсистемами – это переменная взаимосвязи zi(t), которые, вместе с (4.3.15)-(4.3.16), образуют ограничение связи. Эта формулировка называется глобальной и обозначается SG. Можно сделать следующее допущение. Глобальная проблема SG заменяется группой N подзадач, соединенных вместе через вектор параметров a=(a1,…,aN) и обозначенных si(a), i=1,…,N. Другими словами, глобальная системная задача SG включена в группу подсистемных проблем si(a) через внутренний параметр (Sandell и др., 1978) таким образом, что для определенного значения a\*, подсистемы Si(a\*), и i=1,…,N, дают желаемое решение для SG. Используя обозначения иерархического управления, эта внутренняя идея это и есть понятие согласования, но используя терминологию математического программирования задач, она называется основной проблемой (Geoffrion, 1970). На рисунке 4.6 изображена двухуровневая структура управления большой системой. Под этой стратегией, на i-й итерации каждый местный контроллер i получает от координатора (второй уровень иерархии), решает и передает (сообщает) некоторую функцию этого решения координатору.

Координатор, в свою очередь, оценивает следующее значение , т.е.:

 (4.3.19)

где ei – это l-й размер шага итерации, и новый компонент dl, как мы вскоре увидим, часто берется за функцию ошибка взаимодействия:

 (4.3.20)

Внутреннюю переменную взаимодействия zi(\*) в (4.3.20) можно считать частью управляющей переменной доступной для контроллера i, в этом случае вектор параметра a(t) является набором двойных переменных или множителем Лагранжа, который соответствует ограничениям уравнения взаимодействия (4.3.16). Фундаментальная идея, которая стоит за этим подходом должна преобразовать задачу поиска минимума первоначальной системы в более легкую задачу поиска максимума, решение которой можно получить посредством двухуровневой итеративной схемы. Которая обсуждалась выше.

Введем двойную функцию

 (4.3.21)

к объекту (4.3.15), где Лагранжиан L(\*) определен как:

 (4.3.22)

где вектор параметра а состоит из k множителей Лагранжа. Таким образом, первоначально ограниченная (взаимодействием подсистем) оптимизационная задача превращается в неограниченную, другими словами ограничение (4.3.16) удовлетворяется через определение набора множителей Лагранжа ai, i=1,…,k. В таких случаях, когда функции ограничений выпуклые, теорема сильной двойственности Лагранжа (Geoffrion, 1971a, b; Singh, 1980) показывает, что

 (4.3.23)

определяя, что минимизация J в (4.3.17) для объекта (4.3.15)-(4.3.16) эквивалентна максимуму двойной функции q(a) в (4.3.21) по параметру a. Чтобы облегчить решение этой задачи, замечено, что для определенного набора этих множителей Лагранжа а=а\*, Лагранжиан можно переписать в виде:

(4.3.24)

который обнаруживает, что декомпозицию применяют к Лагранжиану таким образом, что, подлагранжиан Li существует для каждой подсистемы. Каждая подсистема будет стремиться минимизировать свой собственный подлагранжиан Li, как определенно в (4.3.24) для объекта (4.3.15) и используя множители Лагранжа a\*, которые считаются известными функциями на первом уровне иерархии. Результат каждой такой минимизации позволит определить двойственную функцию q(a\*) в (4.3.21). На втором уровне, на котором решение всех подсистем первого уровня известны, значение q(a\*) будет изменено типичной неограниченной оптимизацией, например метод Ньютона, градиента или скоростного градиента. Градиентные методы используются потому, что градиент q(a) определяется:

 (4.3.25)

это ошибки взаимодействия подсистем, которые известны из решений первого уровня и определяет градиент f по х. На втором уровне вектор a изменяется по формуле (4.3.19) и рисунку 4.6. Если применяется градиентный метод (с крутым склоном), вектор dl в (4.3.19) является просто l-й итеративной ошибкой взаимодействия el(t). Однако, для повышения точности вычислений определим скоростной градиент как:

 (4.3.26)

где

 (4.3.27)

и d0=e0. Как только вектор ошибки e(t) достигает нуля, появляется оптимальное иерархическое управление s. Ниже дана пошаговая процедура вычисления для метода согласования цели иерархического управления.

Алгоритм 4.1. Метод согласования цели.

Шаг 1. Для каждой подсистемы первого уровня, минимизируем каждый подлагранжиан Li, используя известный множитель Лагранжа a=a\*, так как подсистемы линейные, может быть использовано уравнение Риккати. Сохраним решение. (Читатели не знакомые с уравнением Риккати могут прочитать раздел 4.3.2, метод прогнозирования взаимодействия).

Шаг 2. На втором уровне используется итеративный метод скоростного градиента, похожий на (4.3.26)-(4.3.27), чтобы изменить траектории a\*(t) как в (4.3.19). Как только общая ошибка взаимодействия системы будет нормализована из

 (4.3.28)

и будет достаточно мала, будет достигнуто оптимальное решение для системы. Здесь – размер шага интегрирования.

Два примера ниже иллюстрируют метод согласования цели или баланса взаимодействия. Первый пример, который был предложен Pearson (1971), и позже рассмотрен Singh (1980) и Jamshidi (1983), использован в изменненой форме. Второй пример показывает модель многоколенной задачи загрязнения реки (Beck, 1974; Singh, 1975). Полная оценка многоуровневых методов дана в секции 4.6, а описание нелинейных многоуровневых нелинейных систем в главе 6. Две альтернативы решения этого иерархического управления основаны на расширенных рядах Тейлора и Чебышева в разделе 4.6.

Пример 4.3.1. Рассмотрим систему 12-го порядка введенную Pearson (1971) и показанную на рис 4.7 с уравнением состояния:

 (4.3.29)

и квадратичной функцией оценки:

 (4.3.30)

с

где

Вектор выхода системы представлен как:

 (4.3.31)

Необходимо найти стратегию иерархического управления по методу баланса взаимодействий (согласования цели).

Решение: Из схемы системы, показанной на рисунке 4.7 (пунктирные линии) и матрицы состояния (4.3.29) ясно, что есть четыре подсистемы третьего порядка соединенных через шесть ограничивающих уравнений (по числу пунктирных линий на рис. 4.7):

 (4.3.32)

где ei, i=1,…,6 представляет ошибки взаимодействия между четырех подсистемами. Задачи подсистем первого уровня были решены через набор из четырех матричных уравнений Риккати третьего порядка:

 (4.3.33)

где Ki(t) – это положительно определенная матрица Риккати ni x ni и . Методы «без взаимодействия» и «удвоения» решают дифференциальное матричное уравнение Риккати, предложены Davison и Maki в 1973 и рассмотрены Jamshidi в 1980,были использованы для компьютерного решения (4.3.33). Уравнения состояния подсистем были решены стандартным методом Рунге-Кутта четвертого порядка, а итерации второго уровня были выполнены по схеме скоростного градиента (4.3.19), (4.3.26)-(4.3.27), используя кубическую сплайн интерполяцию (Hewlett-Packard, 1979) для оценки подходящих численных интегралов. Размер шага был выбран =0.1, как и в более ранних рассмотрениях этого примера (Pearson, 1971; Singh, 1980). Алгоритм скоростного градиента позволил уменьшить ошибку с 1 до за шесть итераций, как показано на рисунке 4.8, который был в тесной связи с результатами предыдущих исследований модифицированной версии системы (4.3.29), полученными Singh (1980). Рассмотрим второй пример.

Пример 4.3.2. Рассмотрим двухколенную модель задачи управления загрязнением реки.

 (4.3.34)

где каждое колено (подсистема) реки имеет два состояния – x1 – это концентрация биохимической потребности в кислороде (БПК) (биохимическая потребность в кислороде представляет собой уровень содержания кислорода полученного в результате распада органического вещества) и х2 – это концентрация растворенного кислорода (РК) – и управление u1 – это БПК вод втекающих в реку. Для квадратичной функции оценки

 (4.3.35)

С Q=diag(2,4,2,4) и R=diag(2,2), необходимо найти оптимальное управление, которое оптимизирует (4.3.35) для объекта (4.3.34) при x(0)=(11 -11)T.

Решение: Как видно из (4.3.34)-(4.3.35), две задачи первого уровня идентичны, и матричное уравнение Риккати второго порядка решается интегрированием (4.3.33) используя метод Рунге-Кутта четвертого порядка при =0.1. Ошибка взаимодействия в этом примере снижена до за 15 итераций, как показано на рисунке 4.9. Оптимальные концентрации БПК и РК двух колен реки показаны на рисунке 4.10.

**4.3.2. Метод прогнозирования взаимодействия.**

Альтернативный подход к оптимальному управлению иерархическими системами, который имеет как открытый, так и закрытый контур управления, - это метод прогнозирования взаимодействия, который основывается на работе Takahara (1965), который избегает упоминания о градиентных итерациях второго уровня. Рассмотрим большую линейную взаимосвязанную систему, которая декомпозирована на N подсистем, каждая из которых может быть описана

 (4.3.36)

Где вектор взаимодействия zi:

 (4.3.37)

Задача оптимального управления на первом уровне – найти управление ui(t), которое удовлетворяет (4.3.36)-(4.3.37), минимизируя обычную квадратичную функцию оценки:

 (4.3.38)

Эту задачу можно решить введением множества множителей Лагранжа ai(t), и векторов косостояния pi(t), чтобы увеличить ограничение уравнения взаимодействия (4.3.37) и подсистем динамического ограничения (4.3.36) до подынтегральной функции оценки, т.е. Гамильтониан i-й подсистемы будет определен как:

 (4.3.39)

Затем должно быть написано несколько необходимых условий:

 (4.3.40)

 (4.3.41)

 (4.3.42)

 (4.3.43)

где векторы ai(t) и zi(t) – уже не считаются неизвестными на первом уровне, и фактически ai(t) увеличивает zi(t), чтобы образовать широкоразмерный вектор согласования, который мы рассмотрим ниже. Для решения задачи первого уровня, надо принять как известную. Замете, что ui(t) можно выделить из (4.3.43):

 (4.3.44)

и подставить в (4.3.40)-(4.3.42), получив:

 (4.3.45)

 (4.3.46)

который образует линейную двухточечную краевую (ДТК) задачу, и, как в (4.3.33) . Можно увидеть, что ДТК задача может быть разложена введением матрицы Риккати. Это выглядит как:

 (4.3.47)

где gi(t) – это разомкнутый сопряженный или компенсирующий вектор, размерностью ni. Если обе части уравнения (4.3.47) продифференцированы и и из (4.3.46) и (4.3.45) подставлены в него, можно вновь использовать (4.3.47) и уравнительные коэффициенты для первого и нулевого порядка xi(t), получив следующие матричные и векторные дифференциальные уравнения:

 (4.3.48)

 (4.3.49)

где конечные условия Ki(tf) и gi(tf) вытекают из (4.3.41) и (4.3.47).

 (4.3.50)

В результате данного уравнения оптимальное уравнение первого уровня становится

 (4.3.51)

который имеет частичную закрытую обратную связь и прямую (открытую) обратную связь. Можно сделать два вывода. Первый, решение дифференциального, симметричного матричного уравнения Риккати, в которое включены ni(ni+1)/2 нелинейных скалярных уравнений не зависит от первоначального состояния xi(0). Второй, в отличие от Ki(t), gi(t) в (4.3.49) посредством zi(t) зависит от xi(0). Это свойство будет использовано в разделе 4.4, чтобы получить абсолютно закрытое управление в иерархической структуре.

Задача второго уровня сильно изменяет новый вектор согласования . Для этой цели определите аддитивно отделяемый Лагранжиан:

 (4.3.52)

Значение ai(t) и zi(t) можно получить из:

 (4.3.53)

 (4.3.54)

т.е.:

 (4.3.55)

Процедура согласования второго уровня на итерации (l+1) имеет вид:

 (4.3.56)

Метод прогнозирования взаимодействия формулируется следующим алгоритмом:

Алгоритм 4.2 Метод прогнозирования взаимодействия для непрерывных систем:

Шаг 1. Решить N независимых дифференциальных матричных уравнений Риккати (4.3.48) с конечным условием (4.3.50) и сохраните Ki(t), i=1,2…,N. Инициализируйте ai(t) случайными числами и найдите соответствующее значение для zi(t).

Шаг 2. На l-й итерации используйте значения чтобы решить сопряженное уравнение (4.3.49), с конечным условием (4.3.50). Сохраните gi(t), i=1,2,…,N.

Шаг 3. Решите уравнение состояния

 (4.3.57)

И сохраните xi(t), i=1,2,…,N.

Шаг 4. На втором уровне используйте результаты шагов 2 и 3 и (4.3.56) чтобы изменить согласующий вектор:

Шаг 5. Проверьте сходимость на втором уровне, оценив общую ошибку взаимодействия:

 (4.3.58)

Шаг 6. Если необходимая сходимость достигнута – остановитесь. Иначе, установите l=l+1 и перейдите к шагу 2.

Важно отметить, что в зависимости от типа цифрового компьютера, и его операционной системы, расчеты подсистем могут осуществляться параллельно, а также N-матричное уравнение Риккати на шаге 1 не зависит от xi(0), и значит их необходимо вычислить один раз, не зависимо от числа итераций второго уровня в алгоритме прогнозирования взаимодействия (4.3.56). В отличие от методов согласования цели, zi(t) не нужен в функции оценки, который был необходим, чтобы избежать однородности, о чем будет написано в следующем разделе.

Метод прогнозирования взаимодействия, введенный Tokahara (1965), был рассмотрен многими исследователями, которые внесли в него существенный вклад. Среди них Titli (1972) который назвал этот метод смешанным (Singh, 1980) и Cohen и др. (1974), который предоставил более убедительные доказательства сходимости чем предложенные ранее. Smith и Sage (1973) рассмотрели эту схему для нелинейных систем, которая будет рассмотрена в Главе 6. Сравнение методов прогнозирования взаимодействия, согласования цели и подходов без интеграции, рассмотренных в разделе 4.4, дано в разделе 4.5. Следующие два примера, а потом пример в САПР иллюстрирует метод прогнозирования взаимодействий.

Пример 4.3.3. Рассмотрим систему четвертого порядка

 (4.3.59)

Где x(0)=(-1,0.1,1.0,-0.5)T, квадратичная функция оценки Q=daig(2,1,1,2), R=diag(1,2) и нет граничного штрафа. Надо использовать метод прогнозирования взаимодействия и найти оптимальное управление для tf =1.

Решение: Систему разделили на две подсистемы второго порядка и применили методы, описанные в алгоритме 4.2. На первом шаге решили два независимых дифференциальных матричных уравнения Риккати используя как дублирующий алгоритм Davison и Maki (1973), так и стандартный метод Рунге-Кутта. Элементы матрицы Риккати были представлены в виде квадратичного полинома в ряде Чебышева (Newhouse,1962), для удобства вычислений:

 (4.3.60)

На первом уровне были решены два сопряженных уравнения второго порядка в виде (4.3.49) и два уравнения состояния подсистем, как показано в алгоритме 4.2 в шаге 3, используя метод четвертого порядка Рунге-Кутта и первоначальные значения

 (4.3.61)

На втором уровне векторы взаимодействия [a11(t),a12(t),z11(t),z12(t)] и [a21(t),a22(t),z21(t),z22(t)]T были спрогнозированы с использованием рекурсивных отношений (4.3.56), и на каждой итерации производился обмен информацией с подсчетом общей ошибки взаимодействия (4.3.58) для и программы кубической сплайн интерполяции. Ошибку взаимодействия снизили до за шесть итераций, как показано на рисунке 4.11. Были получены оптимальные значения выхода для Ci =(1 1) и сигнала управления. Затем для сравнения первоначальную систему (4.3.59) оптимизировали, решив нестационарное матричное уравнение Риккати четвертого порядка обратным интегрированием, и для хi(t), i=1,2,3,4; yj(t) и uj(t), j=1,2. Значения выхода и сигналы управления как для случая иерархического управления, так и для централизованного, показаны на рисунке 4.12. Отметьте относительно точное соответствие между значениями выхода для первоначальной соединенной и иерархической разъединенной систем. Но как и ожидалось, эти два уравнения различны.

Теперь рассмотрим второй пример.

Пример 4.3.4. Рассмотрим систему восьмого порядка

Необходимо использовать метод прогнозирования взаимодействия для нахождения u\*.

Решение: Система была разложена на две подсистемы четвертого порядка и были выбраны tf=2, =0.1 , Q1=Q2=I4, R1=R2=1. Первоначальные значения , i=1,2 и состояние х0 были приняты за , и . Сходимость была очень быстрой, как видно на рисунке 4.13. Всего за четыре итерации второго уровня ошибка взаимодействия была снижена до . Фактически была быстрая сходимость для различных x0 и .

САПР пример 4.3.1. Рассмотрим систему четвертого порядка в примере 4.3.1 в (4.3.59):

Где x(0)=(-1,0.1,1.0,-0.5)T , квадратичная функция оценки Q =diag(2,1,1,2), R=diag(1,2) и нет граничного штрафа. Необходимо использовать LSSPAK или подобное программное обеспечение и метод прогнозирования взаимодействия и найти оптимальное управление для tf=2.

Решение: Как и раньше, система делится на две подсистемы второго порядка, и уравнения Риккати для подсистем решаются с использованием RICRKUT от LSSPAK/PC, а их решения представлены в виде полинома четвертого порядка для удобства вычислений. Используя программу INTRPRD от LSSPAK/PC реализуют алгоритм прогнозирования взаимодействия и схождение достигается за пять итераций. Точные выборки из выполнения этого САПр примера приведены ниже. Инструкции для вычерчивания программы прогнозирования взаимодействия появляются, когда на экране появится чертеж; нажмите Enter, чтобы вернуться к меню.

Если вы хотите вывести чертежи через принтер откройте DOS файл GRAPHICS до запуска программы, когда вы захотите вывести чертеж, нажмите shift-PrtScr.

Optimization via the interaction prediction method.

Initial time (to): 0

Final time (tf): 2

Step size (Dt): .1

Total no. of 2nd level iterations = 6

Error tolerance for multi-level iterations - .00001

Order of overall large scale system = 4

Order of overall control vector (r) = 2

Number of subsystems in large scale system = 2

Matrix Subsystem state orders-n sub i 0.200D+01 0.200D+01

Matrix Subsystem input orders-r sub i 0.100D+01 0.100D+01

Polynomial approximation for the Ricatti matrices to be used.

Matrix Ricatti coefficients for SS# 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.453D+01 | -.259D+01 | 0.794D+01 | -762D+01 | O.186D+01 |
| 0.978D-01 | -.793D-01 | 0.252D+00 | .233D+00 | 0.571D-01 |
| 0.490D+00  |  0.759D-02  | -.109D+00  | 0.975D-01  | -.531D-01 |

Matrix Ricatti coefficients for SS# 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.112D+01 | -.815D+01 | 0.361D+01 | 0.455D+01 | 0.105D+01 |
| -0.149D+00 | -.322D-01 | 0.697D-01 | .284D-01 | 0.183D-01 |
| 0.815D+00  |  0.642D-01  | -.295D+00  | 0.305D+00  | -.138D+00 |

System Matrix A

|  |  |  |  |
| --- | --- | --- | --- |
| 0.200D+01 | 0.100D+00 | 0.100D-01 | 0.000D+00 |
| 0.200D+00 | 0.100D+01 | 0.100D+00 | 0.500D+00 |
| 0.500D-01 | 0.150D+00 | 0.100D+01 | 0.500D-01 |
| 0.000D+00 | -0.200D+00 | 0.250D+00 | -0.120D+01 |

Matrix Input Matrix B

|  |  |
| --- | --- |
| 0.100D+01 | O.OOOD+00 |
| 0.100D+00 | O.OOOD+00 |
| O.OOOD+00  | O.250D+O0 |

Matrix Input Cost Function R

|  |  |
| --- | --- |
| 0.100D+01 | O.OOOD+OO |
| 0.000D+O0  | 0.200D+01 |

Matrix Lagrange Multiplier Initial Values

|  |
| --- |
| 0.100D+01 |
| O.IOOD+Ol |
| 0.100D+01 |
| 0.100D+01 |

Matrix Initial conditions vector xO

|  |
| --- |
| -.100D+01 |
| 0.100D+00 |
| 0.100D+01 |
| -.500D+00 |

Subsystem no. 1 at 2nd level iteration no. 1

Subsystem no. 2 at 2nd level iteration no. 1

At second level iteration no. 1 interaction error = 0.347D+00

Subsystem no. 1 at 2nd level iteration no. 2

Subsystem no. 2 at 2nd level iteration no. 2

At second level iteration no. 2 interaction error = 0.771D - 03

Subsystem no. 1 at 2nd level iteration no. 3

Subsystem no. 2 at 2nd level iteration no. 3

At second level iteration no. 3 interaction error = 0.507D - 03

Subsystem no. 1 at 2nd level iteration no. 4

Subsystem no. 2 at 2nd level iteration no. 4

At second level iteration no. 4 interaction error = 0.323D - 04

Subsystem no. 1 at 2nd level iteration no. 5

Subsystem no. 2 at 2nd level iteration no. 5

At second level iteration no. 5 interaction error = 0.310D - 05

Оптимальные отклики показаны на рисунке 4.14, а схождение на рисунке 4.15.

Другие применения метода прогнозирования взаимодействия представлены в разделе задач.

**4.3.3 Согласование цели и однородность**

Когда в (4.3.15) – (4.3.17) обсуждался метод согласования цели, было замечено, что положительно определенные матрицы Si были введены в функцию оценки (4.3.17) для того, чтобы избежать однородности. Чтобы убедится в этом, обратимся к задаче минимизации Li в (4.1.24) для объекта (4.3.15). Пусть i-й Гамильтониан подсистемы имеет вид:

 (4.3.62)

Одно из необходимых уравнений для решения задачи i-й подсистемы на первом уровне

 (4.3.63)

или

 (4.3.64)

где однородное решение появляется, если не появляется в функции оценки. Чтобы избежать однородности на первом уровне возможны два альтернативных подхода. Следующий пример иллюстрирует два подхода.

Пример 4.3.5. Рассмотри следующую систему:

 (4.3.65)

Необходимо найти (u1,u2), такие, чтобы они удовлетворяли (4.3.65), а квадратичная функция оценки

 (4.3.66)

минимизировалась методом согласования цели.

Решение: Из (4.3.65) –(4.3.66) ясно, что систему можно разделить на две подсистемы первого порядка.

 (4.3.67)

 (4.3.68)

с ограничением взаимодействия

 (4.3.69)

Задача в настоящий момент имеет следующий Гамильтониан:

(4.3.70)

в котором переменная взаимодействия появляется линейно. Применение метода согласования цели для данной формулировки приведет к однородности, так как z1 появляется линейно в (4.3.70). Следующая системная переформулировка задач поможет избежать однородности.

Часть а – подсистема 1, переменные состояния

Часть б – подсистема 1, переменные управления

Часть в – подсистема 2, переменные состояния

Часть г – подсистема 2, переменные управления

**4.3.3.а. Переформулировка 1.**

Bauman (1968) предложил переписать ограничения взаимодействия квадратичной формы

 (4.3.71)

которая даст следующее необходимое условие для оптимизации на первом уровне:

 (4.3.72)

для первой подсистемы и

 (4.3.73)

для второй подсистемы. После введения формулы Риккати (4.3.72) и (4.3.73) мы получим:

и

где ki(t) – i-я скалярная нестационарная матрица Риккати для подсистемы. Согласование на втором уровне достигается через следующие итерации:

Эта переформулировка помогает избежать однородности, но делает схождение итераций второго уровня очень медленным.

**4.3.3.б. Переформулировка 2.**

Singh (1980) предложил альтернативную формулировку, которая не только позволит избежать однородности, но и даст хорошее схождение процедура основывается на том, чтобы найти х через вектор взаимодействия z и подставить его в функцию оценки, т.е. z можно представить как:

где G – считается неоднородной и переформулированный Гамильтониан представлен в виде:

В этом примере матрица G – однородна, но решение можно получить. Гамильтониан имеет вид:

А задача подсистемы первого уровня имеет вид

и

вторую подсистему можно решить сразу же, так как уравнение p2 – косостояние отделено от х2 и может быть решено в обратном порядке и подставлено в уравнение х2, что приведет к тому, что решение уравнения Риккати в данном примере не требуется. Но для первой подсистемы, исходя из формулировки задач первого уровня в прогнозировании взаимодействия (4.3.40) – (4.3.51), необходимо как уравнение Риккати, так и открытое сопряженное (компенсирующее) векторное уравнение. Для этого примера задача первой подсистемы имеет вид

где два дифференциальных уравнения для ki(t) и gi(t) нужно решить в обратном порядке. В то время как для второй подсистемы не нужно решать вспомогательное уравнение, надо решить два таких уравнения для первой подсистемы. В общем эта переформулировка требует решения

 (4.3.74)

что означает, что уравнения вектора косостояния p отделено от х и может быть решено в обратном порядке (без решения уравнения Риккати) и подставлено в верхнее уравнение для нахождения х. Так как матрицы A, B, Q и R – блок-диагональные, задачу (4.3.74) можно разделить на N задач подсистем с условием, что отделяемо от z, где V=G-1.

