ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 56 за 2008 г.

Иванов Д.С., Вальтер Т.,
Биндель Д., Овчинников М.Ю.

Стенд для отработки алгоритмов управления движением многоэлементных систем

Рекомендуемая форма библиографической ссылки: Стенд для отработки алгоритмов управления движением многоэлементных систем / Д.С.Иванов [и др.] // Препринты ИПМ им. М.В.Келдыша. 2008. № 56. 32 с. URL: http://library.keldysh.ru/preprint.asp?id=2008-56

Стенд для отработки алгоритмов управления движением многоэлементных систем. Д.С. Иванов, Т. Вальтер, Д. Биндель, М.Ю.Овчинников. Препринт ИПМ им.М.В.Келдыша РАН, Москва, 32 страницы, 27 рисунков, библиография 10 наименований.

Приведено описание макета, созданного в Центре космической техники и микрогравитации (ZARM) Бременского университета (Германия) для имитации движения многозвенных систем. Приведены расчеты параметров отдельных элементов макета. Дано описание алгоритма управления его импульсными двигателями. Приведены расчеты для фильтра Калмана, использующегося в задаче определения ориентации макета.

Ключевые слова: импульсный двигатель, датчик угловой скорости, оптимальное управление, фильтр Калмана,

Laboratory Facility to Verify Motion Control Algorithms for Multi-element Systems, D.S. Ivanov, T. Walter, D. Bindel, M.Yu. Ovchinnikov, Preprint of KIAM RAS, Moscow, 32 Pages, 27 Figures, 10 References.

Laboratory facility for simulation of planar formation mock-up dynamics is being developed at Zentrum für Angewandte Raumfahrttechnologie und Mikrogravitation (ZARM) in Bremen (Germany) which is intended to be used for verification of motion control algorithms. Mock-up parameters are determined and elements of the optimal control theory are stated. The control algorithm together with Kalman filter parameters which are used to control the mock-up are developed.

Key words: impact thruster, angular velocity sensor, optimal control, Kalman filtering

1. Введение

Перед запуском искусственный спутник Земли подвергают многоэтапным проверкам и испытаниям. Эти испытания дают важную информацию о его будущем поведении на орбите, позволяют выявить ошибки, допущенные при его разработке и изготовлении, помогают найти границы применимости того или иного метода управления и многое другое, на основе чего можно надеяться на четкую и надежную работу спутника на орбите. Однако силы и моменты, которые испытывает спутник на орбите, значительно отличаются от тех, которые действуют на него на Земле. Это затрудняет проведение его динамических испытаний в лабораторных условиях. Тем не менее, существуют способы проведения испытаний спутников, которые в определённой мере имитируют условия на орбите, позволяя таким образом отработать алгоритмы идентификации и управления углового движения тела в космосе.

Рис. 1.1. МКС

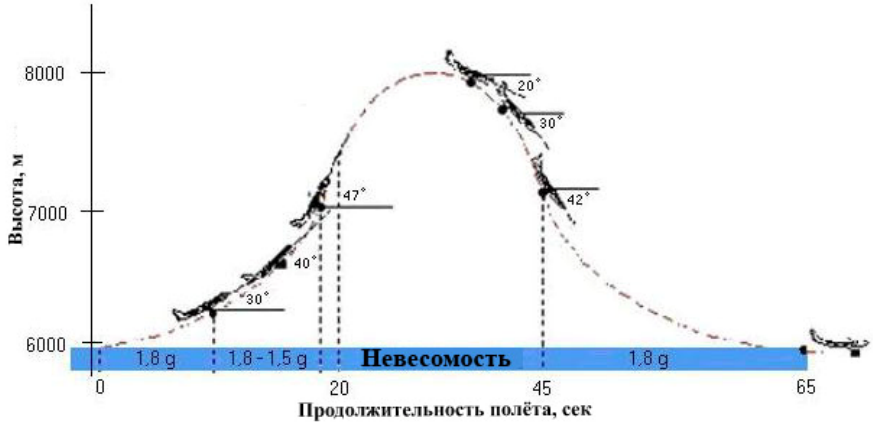

Рис. 1.2. Невесомость в самолёте

Пожалуй, лучшие условия для испытаний достигаются на Международной космической станции (рис.1.1), где спутник может иметь сразу 6 степеней свободы. Однако стоимость доставки груза на МКС весьма высока и составляет около 20 тыс. евро за килограмм. Другой способ провести испытания – это поместить испытуемый спутник или его макет в самолет. На параболическом участке траектории самолёта все тела, находящиеся в самолёте, будут находиться в условиях микрогравитации (рис. 1.2). Недостаток этого способа заключается в том, что он также достаточно дорогой, и, кроме того, позволяет проводить эксперименты в течение очень короткого промежутка времени, за который сложно получить какие-либо результаты по отработке алгоритмов движения спутника. Ещё один

Рис. 1.3. Drop Tower

способ проведения подобных экспериментов – это использование Drop Tower, расположенной в Центре прикладных космических технологий и микрогравитации в Бремене (рис. 1.3) - башни, откуда откачан воздух, что позволяет падающему в ней телу в течение почти 10 секунд находиться в условиях микрогравитации. Но, опять же, время проведения экспериментов мало, а стоит “бросок” тела весьма дорого.

Возникает задача проведения наземных испытаний спутника и используемых алгоритмов управления, относительно недорогих, может, не в полной мере соответствующих условиям на орбите, но неограниченных по времени проведения экспериментов. Один из способов проведения таких испытаний – это использование скользящего тела, которое испытывает слабое трения при движении по плоскости. Такое скользящее тело может беспрепятственно совершать движения на гладкой поверхности и менять ориентацию относительно оси, перпендикулярной поверхности.

Для этой цели в Центре прикладных космических технологий и микрогравитации (Zentrum für angewandte Raumfahrttechnologie und Mikrogravitation - ZARM) при Университете города Бремена (Германия) был разработан стенд под названием LuVeX (**Lu**ftkissen **V**ehikel **Ex**periment), на котором можно исследовать динамику движения спутников. На настоящий момент создано два макета, передвигающиеся благодаря воздушной подушке, приводимые в движение импульсными двигателями на сжатом воздухе. Построив алгоритмы управления этими двигателями, можно провести эксперименты по управлению положением и ориентацией макетов. Эти результаты будут полезны для разработки алгоритмов управления реальными спутниками и, например, управления групповыми полётами типа Formation Flying.

1.1. Ранее разработанные системы

В настоящее время в США проводится много экспериментов для исследования динамики движения спутников и отработки алгоритмов их движения. Такие эксперименты имеют место в индустрии, научной и военной сферах. Рассмотрим некоторые их них.

Одним из экспериментов является SPHERES (рис. 1.4), проект Массачусетского технологического института [1]. Испытания проводятся для отработки

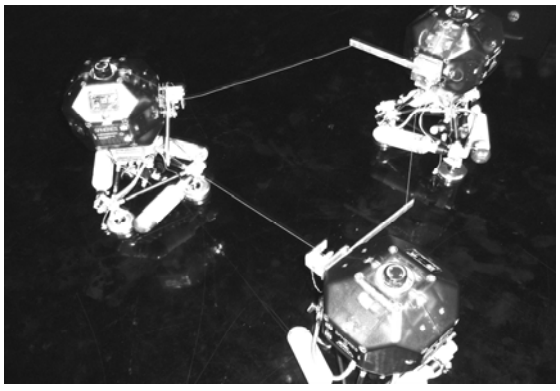

Рис. 1.4. SPHERES

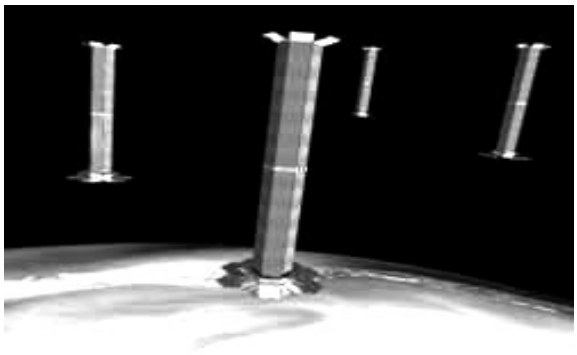

Рис. 1.5. Проект AFRL

движения группы спутников на параболической траектории самолёта и на МКС. Группа состоит из трех наноспутников, которые совершают движения в состоянии невесомости. Такой эксперимент является весьма дорогим и требует больших затрат времени и денег на подготовительном этапе. Поэтому для этих наноспутников построили специальную подставку на воздушной подушке, как изображено на рис.1.4, и предварительные эксперименты и отработку проводили на гладкой поверхности.

Также в США были разработаны наноспутники в Air Force Research Laboratory для военных целей, которые могут с достаточно высоким разрешением производить мониторинг поверхности Земли (рис.1.5) [2]. Однако эксперименты не продвинулись дальше моделирования движения такой группы спутников.

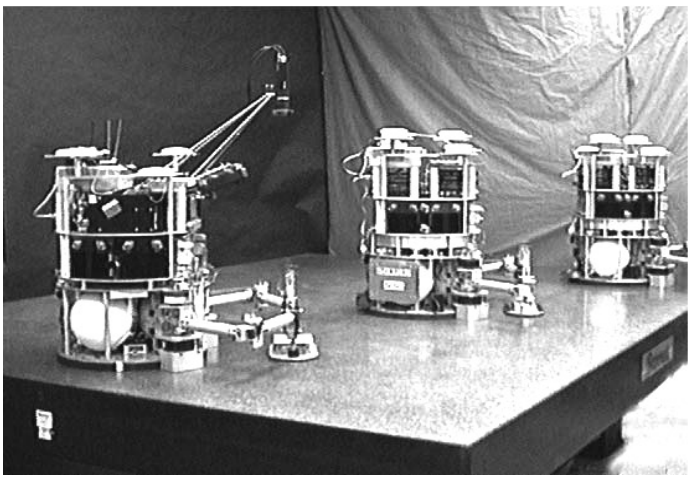

Рис. 1.6. Роботы из Стефордского университета

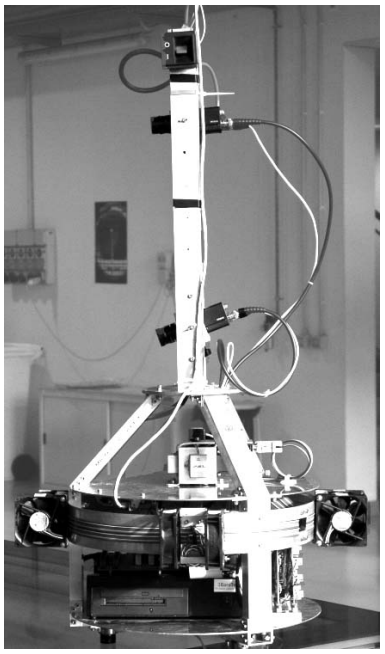

Рис. 1.7. “Smart Flyer”

В Стенфордском университете разработаны три робота, способные скользить по гладкой поверхности на воздушной подушке (рис.1.6) [3]. Этот проект весьма экономичен и не требует больших затрат времени для подготовки проведения экспериментов. Создание таких роботов позволяет проводить эксперименты по исследованию динамики и по отработке алгоритмов управления сложными многозвенными системами.

В Милане в Политехническом университете занимаются созданием робота, названного Smart Flyer, использующегося для отработки алгоритмов управления движением (рис.1.7) [4]. Этот робот оснащен двумя пропеллерами, позволяющими имитировать воздействия импульсных двигателей.

LuVeX является одним из стендов, на котором моделируется движение группы спутников. Он стоит относительно недорого и удобен при проведении экспериментов. Макет имеет небольшую массу, относительно небольшой и экономичный. Для его функционирования требуется лишь ровная поверхность, сжатый воздух и электроэнергия.

Стенд состоит из стола со стеклянным покрытием и двух макетов, которые будем условно называть *старый* (рис. 1.8, справа) и *новый* (рис. 1.8, слева). На момент выполнения настоящей работы новый макет находился в стадии сборки и не полностью функционировал. И именно ему посвящена настоящая работа. При проектировании новой версии LuVeX'a были учтены все недоработки старой версии - были установлены более совершенные импульсные двигатели, были использованы более мощные элементы питания, один большой баллон со сжатым воздухом был заменён четырьмя меньшего размера, вследствие чего макет стал более компактным. После завершения сборки нового макета планируется проведение экспериментов по отработке алгоритмов движения многозвенных систем.

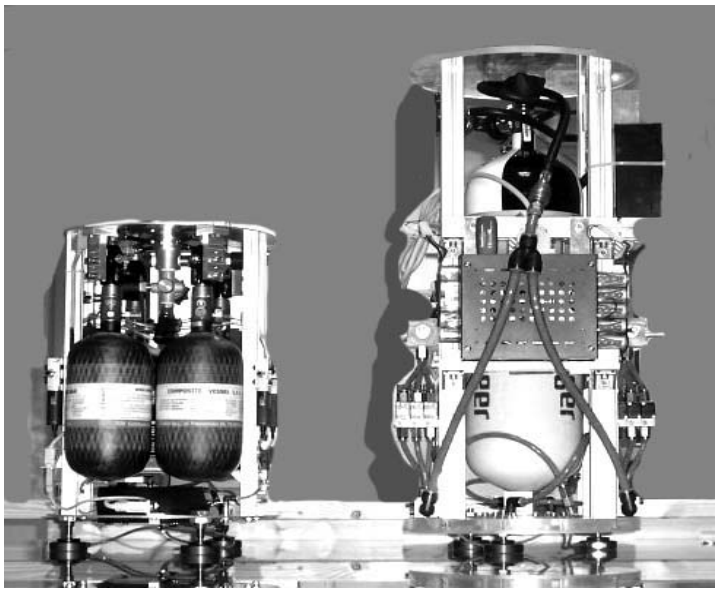

Рис. 1.8. *Новый* (слева) и *старый* макеты Lu-

1.2. Цели создания стенда

Перечислим основные цели, для которых создается стенд [5].

- Отработка алгоритмов управления ориентацией и положением спутников с помощью импульсных двигателей.
- Разработка алгоритмов определения ориентации спутников по снимкам звёздной камеры и с помощью показаний датчика угловой скорости.
- Отработка механизмов передачи информации между спутником и наземной станцией с помощью имитационной схемы: макет – стационарный компьютер.
- Проверка алгоритмов и механизмов взаимодействия нескольких макетов как многозвенной системы, отработка управления группового полёта спутников.
- Обучение на базе стенда специалистов по созданию сложных информационных систем и специалистов по управлению такими системами.

1.3. Требования к стенду

Для достижения заданных целей, макеты должны удовлетворять определенным требованиям относительно конструкции, системы управления, определения ориентации, системы коммуникации, электроснабжения и так далее. Перечислим эти требования.

- Продолжительность автономной работы макета должна составлять 20 – 30 мин.
- Снабжение импульсных двигателей сжатым воздухом должно происходить из резервуаров, расположенных на самом макете.

- На макет необходимо установить бортовой компьютер массой приблизительно 0.75 кг и это нужно учесть при разработке конструкции макета.
- При выборе батарей питания нужно учесть, что бортовой компьютер питается от 6.5 В и потребляет энергию 12 Вт, кроме того остальная часть макета будет потреблять около 10 Вт.
- Требуемое напряжение источника питания не должно превышать 24 В.
- Макет должен беспрепятственно перемещаться и вращаться на плоскости.
- Для вышеупомянутых движений макета нужно установить 6 импульсных двигателей, работающих на сжатом воздухе и регулируемых с помощью клапанов.
- Диаметр макета не должен превышать 0.3 м.
- Из соображений устойчивости в поле тяжести, высота макета не должна быть больше его полутора диаметров.
- Необходимо, чтобы замена блоков питания макета была достаточно быстрой и технологичной.
- Нужно разработать и включить в конструкцию макета систему подключения баллонов со сжатым воздухом.
- Учесть, что полезная нагрузка макета может составлять около 2 кг.
- Суммарная стоимость макета не должна превышать 2 – 3 тысячи евро.

Взаимосвязь всех вышеперечисленных требований можно изобразить на схеме (рис. 1.7).

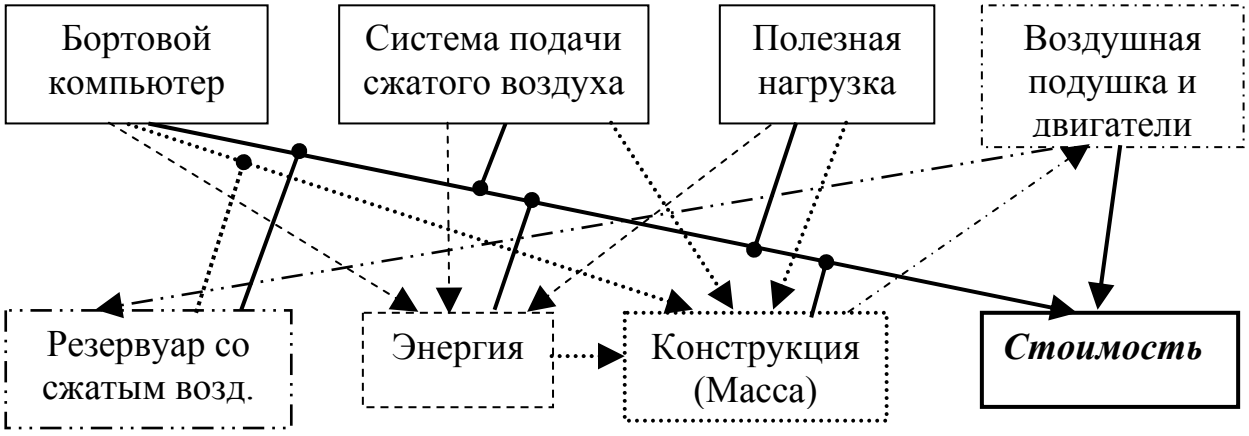

Рис.1.9. Взаимосвязь требований к макету

Как можно увидеть из схемы, все составляющие макета влияют на его конечную стоимость, что является весьма важным параметром, который должен быть обязательно учтен при проектировании макета и при выборе используемых материалов и вариантов сборки макета, но также должны быть выполнены все вышеперечисленные требования относительно свойств и параметров макета. Созданный *новый* макет LuVeX – это макет с оптимальным соотношением цена – требования, где каждая составляющая макета выполняет необходимые функции, но не является при этом дорогостоящей.

2. Устройство макета

Рассмотрим подробно устройство *нового* макета и принципы его функционирования. Проведём расчет параметров для правильной работы отдельных его частей. Методика расчета параметров *нового* макета взята из [5], где рассчитаны эти же параметры для *старого* макета.

2.1. Расчет требуемого объёма воздуха и выбор формы опор макета

На макет действует сила тяжести Земли, вследствие чего вся его масса давит на опоры макета, представляющие собой три небольшие плоскости, а они в свою очередь создают давление на горизонтальную плоскость, на которой стоит сам макет. Понятно, что в таком состоянии макет не может двигаться по плоскости свободно: при движении будет возникать сила трения между ножками-опорами макета (рис.2.1) и плоскостью, что приведёт к торможению макета. Для исключения силы трения используют так называемую «воздушную подушку» – прослойку воздуха между плоскостью ножек и плоскостью стола, которую получают пропусканием воздуха под некоторым напором через пористый материал ножек. Благодаря воздушной подушке тело движется по плоскости свободно, чем фактически имитируется движение тела на орбите, только в проекции на двумерное пространство. Однако для такой воздушной подушки необходимо правильно выбрать материл, из которого будут изготовлены ножки макета, их размеры и подавать на вход в ножки сжатый воздух под определённым давлением.

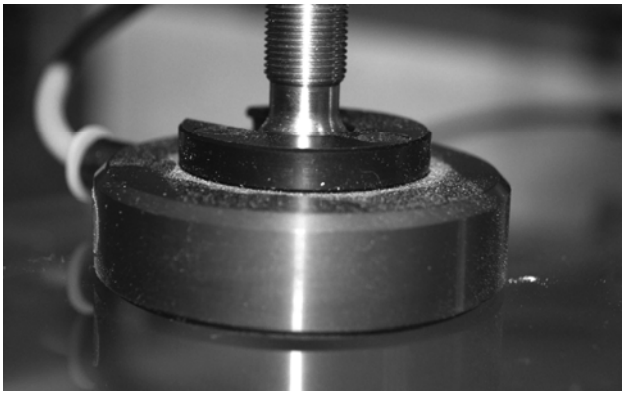

Рис. 2.1. Ножка макета

Для расчета давления, которое необходимо подавать на вход ножкам, необходимо рассчитать расход воздуха на воздушную подушку.

Запишем закон Бернулли сохранения полного давления в потоке газа или жидкости

$$P_{полн.} = P_{стат.} + \rho g h + \rho \frac{u^2}{2} = const\,,$$

где в нашем случае ρ – плотность газа под опорой, h – высота опоры над плоскостью стола, g – гравитационная постоянная, u – скорость истечения газов. В законе Бернулли третье слагаемое ещё называют динамическим давлением

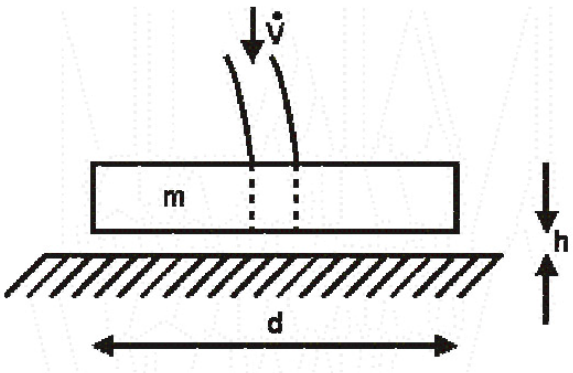

Рис. 2.2. Схема опоры макета

$$P_{дин.} = \rho \frac{u^2}{2}.$$

Статическое давление в нашем случае рассчитывается так:

$$P_{стат.} = \frac{m_{мак.} \cdot g}{n \cdot S_{опоры}}.$$

Здесь $m_{мак.}$ – масса макета, n – число опор макета, $S_{опоры}$ – площадь одной опоры макета.

Так как аппарат находится в равновесии, то статическое и динамическое давления должны быть равны $P_{стат.} = P_{дин.}$ или

$$\frac{m_{мак.} \cdot g}{n \cdot \pi \cdot r^2} = \frac{\rho \cdot u^2}{2}.$$

Отсюда выразим скорость истечения газов u:

$$u = \sqrt{2 \frac{m_{мак.} g}{\rho n S_{опоры}}}.$$

Расход газа вычисляется следующим образом: $\dot{V} = u \cdot n \cdot S_{истеч.}$, где $\dot{V}$ – объемный расход газа, $S_{истеч.}$ – площадь, через которую истекает газ. В нашем случае последнее – это периметр формы опоры A, умноженный на высоту опоры над плоскостью стола h. Таким образом, можно переписать

$$\dot{V} = \sqrt{2 \frac{m_{мак.} g}{\rho n S_{опоры}}} \cdot n \cdot A \cdot h$$

или

$$\dot{V} = \sqrt{\frac{A^2}{S_{опоры}}} \sqrt{2 \frac{m_{мак.} g n}{\rho}} \cdot h. \qquad (2.1)$$

Теперь обоснуем, почему формой опор выбрана именно окружность. Если рассмотреть первый множитель в формуле (2.1), то для круглой формы опор он будет равен:

$$\sqrt{\frac{A^2_{окруж.}}{S_{опоры}}} = \sqrt{\frac{(2\pi r)^2}{\pi r^2}} = 3.54.$$

Для всех других форм опор этот множитель будет больше, а значит, будет больше расход газа. Например, для квадратной формы

$$\sqrt{\frac{A^2_{квадр.}}{S_{опоры}}} = \sqrt{\frac{(4a)^2}{a^2}} = 4.$$

Зная расход газа и время, в течение которого будет тратиться газ, можно рассчитать полный объем газа $V = \dot{V} \cdot t$ или

$$V = \sqrt{\frac{A^2}{S_{опоры}}} \sqrt{2 \frac{m_{мак.} g n}{\rho}} \cdot h \cdot t.$$

Произведём расчет необходимого запаса газа для работы макета в течение $t = 30$ мин $= 1800$ сек, масса *нового* макета $m_{мак.} = 12$ кг, плотность воздуха будем считать $\rho = 1.27$ кг/м^3, толщину воздушной подушки возьмём $h = 10^{-5}$ м, число опор $n = 3$. Тогда минимальный объём, который необходимо затратить при работе системы, составляет $V = 1.5$ м^3.

Однако сжатый воздух будет расходоваться не только на воздушную подушку, но и на работу импульсных двигателей, поэтому назовём полученную цифру *затрачиваемый объём при отсутствии управления*. Понятно, что в общем случае нужно иметь запас воздуха, значительно превышающий этот. Запас воздуха было решено хранить в сжатом виде в 4-х баллонах, объемом 1.1 л каждый (рис.2.3).

Рис. 2.3. Баллон

Рис. 2.4. Размещение баллонов

Эти баллоны симметрично расположены в конструкции (рис.2.4) для того, чтобы центр масс макета находился на линии, соединяющей центры опорных металлических дисков. В баллоны накачивается сжатый воздух под давлением примерно 200 бар, и, таким образом, в четырёх баллонах хранится около 8 м^3 воздуха.

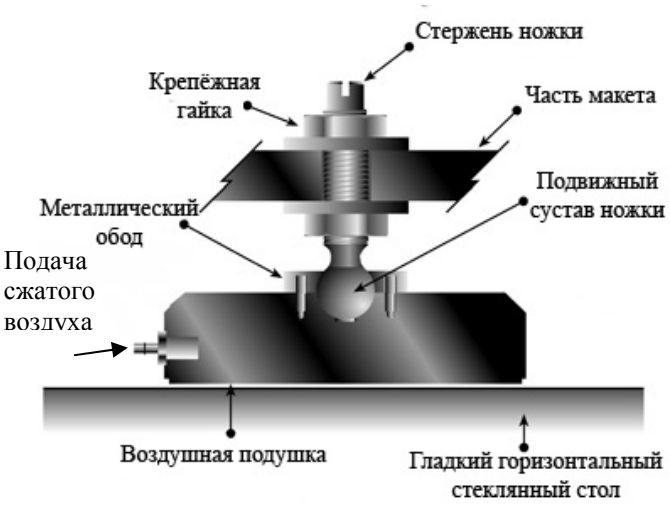

Рис.2.5. Схема ножки макета

Рассмотрим подробнее строение опоры макета, изображенного на рис.2.5. Через канал подачи газа воздух проникает в пористую среду ножки и на выходе из неё создаёт газовую прослойку между ножкой и глад-

кой горизонтальной поверхностью стеклянного стола. Ножки макета также способны двигаться, так как стержень ножки снабжен некоторым суставом, благодаря которому ножка может поворачиваться. Это необходимо для того, чтобы обеспечить касание ножки всей своей поверхностью стола, в случае если он имеет небольшие искривления.

2.2. Импульсные двигатели

Импульсные двигатели, установленные на макете, работают на сжатом воздухе, поступающем из резервуара, соединённого с баллонами. Импульсные двигатели (рис.2.6), состоят из клапана, перекрывающего резервуар со сжатым воздухом, и некоторой трубки, направляющей поток газа. Когда на двигатель поступает команда, клапан убирается, и поток сжатого воздуха течет наружу, передавая тем самым некоторый импульс макету.

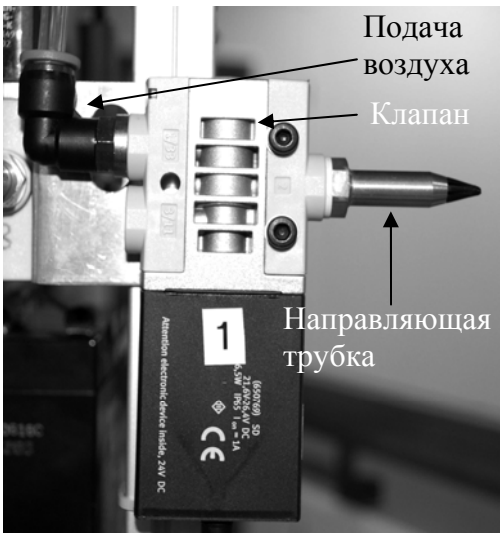

Рис.2.6. Импульсный двигатель

Рассчитаем давление, которое необходимо подавать на вход в импульсный двигатель, чтобы получить наибольший удельный импульс. Сила $\boldsymbol{F}$, действующая на макет со стороны импульсного двигателя, определяется так: $\boldsymbol{F} = m \cdot \boldsymbol{a}$. Здесь m – масса макета, $\boldsymbol{a}$ - ускорение макета за счет действия двигателя. Но с другой стороны эта же сила равна произведению массового расхода двигателя $\dot{m}_{возд.}$ на скорость выходящего потока газа $u_{возд.}$, то есть $\boldsymbol{F}_{тяг.} = \dot{m}_{возд.} \cdot \boldsymbol{u}_{возд.}$. Однако расход двигателя можно расписать так

$$\dot{m}_{возд.} = S_{сопла} u_{возд.} \rho_{возд.},$$

где $S_{сопла}$ – это площадь сечения трубки истечения воздуха (будем называть *площадь сопла*), $u_{возд.}$ и $\rho_{возд.}$ – скорость истечения и плотность воздуха в сопле соответственно. Таким образом, сила тяги, которую дает импульсный двигатель, равна:

$$\boldsymbol{F}_{тяг.} = S_{сопла} \cdot \rho_{возд.} \cdot \boldsymbol{u}^2_{возд.}.$$

Запишем закон Бернулли следующим образом:

$$i_1 + \frac{u_1^2}{2} = i_2 + \frac{u_2^2}{2}.$$

Здесь i – энтальпия идеального газа, а u – скорость течения газа. В случае истечения газа из баллона (из состояния покоя) величиной u_1 можно пренебречь. Тогда

$$u = \sqrt{2(i_1 - i_2)}. \qquad (2.2)$$

Энтальпию идеального газа опишем цепочкой равенств

$$i = U + \frac{P}{\rho} = \frac{1}{\mu} C_V T + \frac{1}{\mu} RT$$

или окончательно

$$i = \frac{1}{\mu} C_P T .$$

Тогда уравнение (2.2) запишется в виде

$$u = \sqrt{\frac{2}{\mu} C_P (T_1 - T_2)} . \qquad (2.3)$$

Используем уравнение адиабаты, из которого следует

$$\frac{P_1^{\gamma-1}}{T_1^{\gamma}} = \frac{P_2^{\gamma-1}}{T_2^{\gamma}} .$$

Отсюда

$$T_2 = T_1 \left(\frac{P_2}{P_1} \right)^{\frac{\gamma-1}{\gamma}} .$$

Тогда уравнение (2.3) преобразуется к виду

$$u = \sqrt{\frac{2}{\mu} C_P T_1 \left[1 - \left(\frac{P_2}{P_1} \right)^{\frac{\gamma-1}{\gamma}} \right]} .$$

Его можно переписать, используя соотношения

$$C_P - C_V = R , \; \frac{C_P}{C_V} = \gamma ,$$

$$u = \sqrt{\frac{2\gamma}{\gamma - 1} RT_1 \left[1 - \left(\frac{P_2}{P_1} \right)^{\frac{\gamma-1}{\gamma}} \right]} . \qquad (2.4)$$

Известно, что максимальный расход газа будет достигаться в критическом (наименьшем) сечении сопла при достижении скорости звука газа в этом сечении. Скорость звука в газе равна

$$u_{звука} = \sqrt{\gamma RT} .$$

В критическом сечении сопла температура газа равна

$$T_{крит.} = \frac{2 \cdot T_1}{1 + \gamma} .$$

Таким образом, подставляем критическую температуру в формулу для скорости звука – скорости, с которой будет истекать газ из сопла. И для этой скорости выразим из уравнения (2.4) значение давления в баллоне P_2, при котором будет максимальный расход газа

$$P_1 = P_2 \left(1 - u_{звука}^2 \frac{\gamma - 1}{2\gamma RT_1} \right)^{\frac{\gamma-1}{\gamma}} . \qquad (2.5)$$

Произведём расчет давления газа P_1 (2.5), которое необходимо подавать на дви-

гатели, при условии, что P_2 – атмосферное давление, $T_1 = 297\,K$, $\gamma = 1.4$. Скорость звука при таких параметрах равна $u_{звука} = 314$ м/с, а $P_2 = 1.89$ бар.

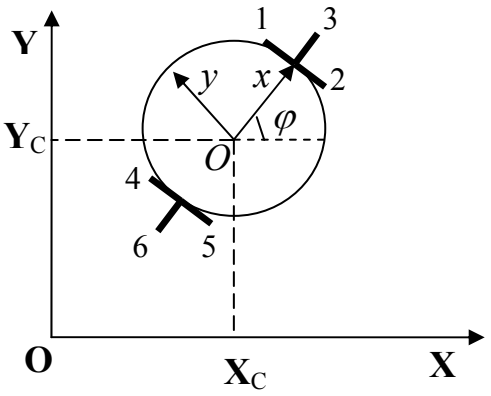

Рис.2.7. Расположение двигателей

Двигатели располагаются на макете таким образом (рис. 2.7), что для создания силы в направлении связанной оси *Ox* и против этого направления действуют двигатели *3* и *6*, для создания силы в направлении оси *Oy* действует сразу пара двигателей *1* и *4*, против направления этой оси *2* и *5*. Для создания вращающего момента по часовой стрелке включаются двигатели *1* и *5*, против часовой стрелки – *2* и *4*.

2.3. Система подачи воздуха

Схема системы подачи воздуха показана на рис. 2.8. Она питается от четырех баллонов с запасенным сжатым воздухом. Давление в них составляет около 200 бар. Однако, как было вычислено выше, для максимального импульса на вход двигателям необходимо подавать давление $P = 2$ бара. В связи с этим был установлен специальный клапан, который обеспечивает постоянство давления воздуха, подаваемого на двигатели и воздушную подушку. Сразу после клапана установлен манометр, с помощью которого можно отслеживать значение давления подаваемого воздуха. Все четыре баллона соединяются посредством резервуара в одну ёмкость, поэтому газ расходуется одновременно из всех баллонов, что позволяет сохранить симметричное распределение масс на макете.

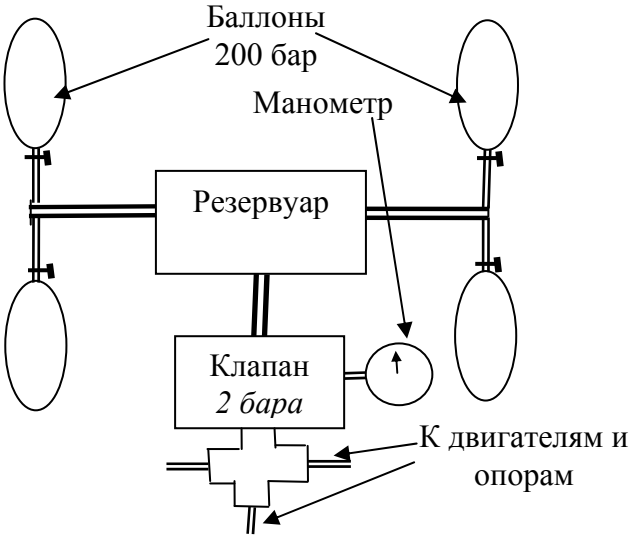

Рис.2.8. Схема системы подачи воздуха

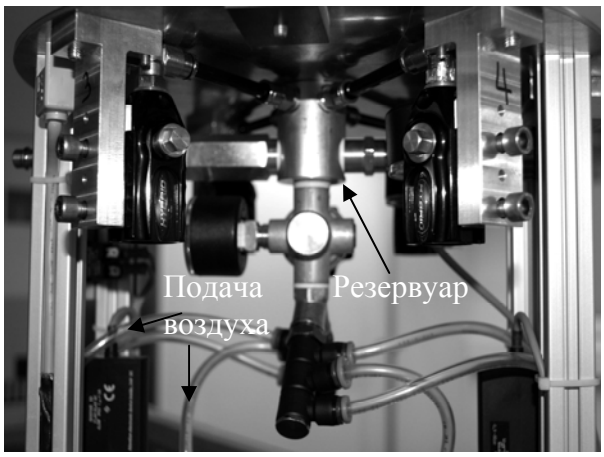

Рис.2.9. Система подачи воздуха

Когда в баллонах заканчивается газ, давление в резервуаре становится меньше 2 бар, поэтому давление, подаваемое на опоры макета, ослабевает, и на

макет начинает действовать сила трения, воздушная подушка исчезает и макет прекращает движение.

2.4. Система электроснабжения

Система электропитания состоит из аккумулятора и преобразователя (рис.2.10). Аккумулятор подает 14 В на преобразователь напряжения, который в свою очередь питает бортовой компьютер, микроконтроллер, управляющий сигналами на импульсные двигатели, датчик угловой скорости, акселерометр и веб-камеру.

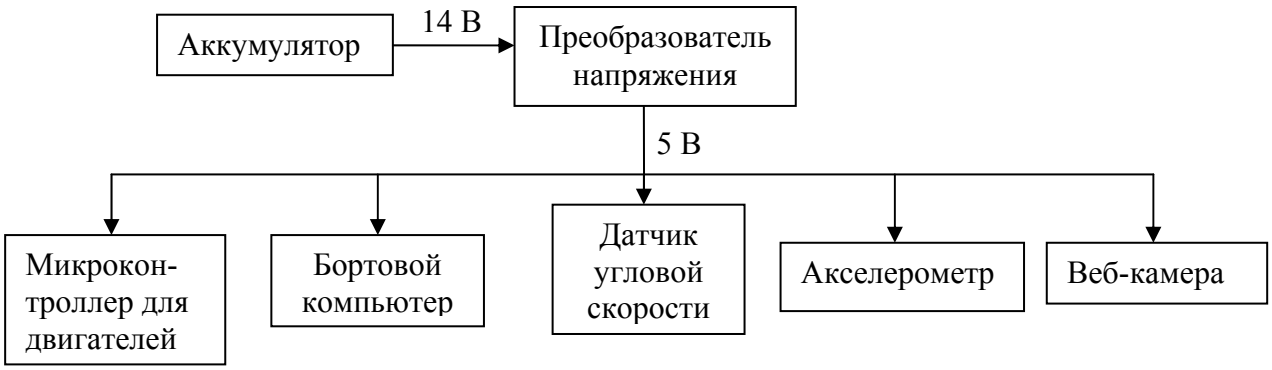

Рис.2.10. Схема системы электропитания

2.5. Система определения ориентации и положения макета и система управления импульсными двигателями

Рис.2.11. Гироскоп

Определения ориентации состоит из двух этапов.

- Обработка на бортовом компьютере показаний датчика угловой скорости (рис.2.11) и акселерометра в реальном времени даёт оценку полного вектора состояния макета.
- Корректировка оценки вектора состояния макета с помощью веб-камеры (рис.2.12) и имитатора звёздного неба – потолка со светящимися светодиодами (рис.2.13). Используя карту положения светодиодов на потолке, по снимку с веб-камеры, установленной на

Рис.2.12. Веб-камера

Рис.2.13. “Звёздное небо”

верхней крышке макета, можно вычислить положение и ориентацию макета, и

тем самым скорректировать оценку вектора состояния макета, полученную с помощью датчика угловой скорости и акселерометра. Однако корректировку можно производить только с некоторым промежутком времени по причине слишком большого времени, необходимого для обработки одного снимка.

Система управления положением и ориентацией макета состоит из бортового компьютера (рис.2.14), микроконтроллера и системы импульсных двигателей. Для импульсных двигателей заложен следующий принцип функционирования: каждые 100 мс компьютер посылает значения на микроконтроллер сколько следующие 100 мс должен работать каждый двигатель. Микроконтроллер в свою очередь подаёт команду открыть клапаны на двигатели. Если управление требует того, чтобы клапан какого-либо двигателя был открыт более 100 мс, компьютер посылает на микроконтроллер значения 100 для этого двигателя столько раз, сколько это необходимо. Бортовой компьютер также снабжен разъемами для подключения монитора, клавиатуры и локальной сети. Это предусмотрено для удобства работы с макетом во время отработки программного обеспечения.

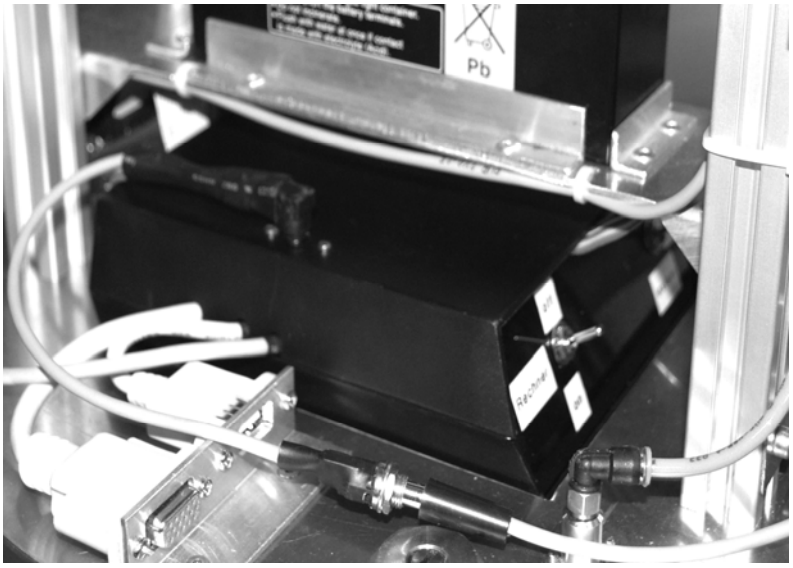

Рис.2.14. Бортовой компьютер

Рис.2.15. Внешний вид макета

Несущая конструкция макета выполнен из лёгкого сплава алюминия и состоит из двух дисков, верхнего и нижнего, и четырёх перекладин, которые надёжно скрепляют эти диски (рис.2.15). К нижнему диску прикреплён бортовой компьютер, к верхнему – четыре баллона с воздухом, а также веб-камера.

3. Реализованный алгоритм управления ориентацией

Рассмотрим алгоритм управления ориентацией рассматриваемого макета. Алгоритм заключается в отслеживании некоторой заданной программной тра-

ектории движения с определённой точностью. При разработке алгоритма использовались элементы теории оптимального управления, а также строился фильтр Калмана, необходимый для устранения шума датчика угловой скорости.

3.1 Элементы теории оптимального управления

Классификация задач оптимального управления

В общем случае задачу управления нельзя ограничивать только достижением некоторого значения вектора состояния $\boldsymbol{x}(t_1)$ [6]. Может оказаться, что в таком строгом достижении этого состояния и нет необходимости: важно, чтобы вектор состояния динамической системы не вышел из некоторой области, определяющей многообразие его допустимых значений. Естественно, каждому заданному закону управления соответствует закон изменения координат вектора состояния, то есть траектория "движения" управляемого объекта в фазовом пространстве. Зачастую процесс управления осуществляется с "ограниченными ресурсами", то есть закон управления не может быть произвольным, а должен выбираться из некоторого множества Ω. Математически задача оптимального управления может быть сформулирована так. Дан управляемый динамический объект, вектор состояния которого подчиняется системе уравнений

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \qquad \boldsymbol{u} \in \Omega. \tag{3.1}$$

Информацию о текущем векторе состояния $\boldsymbol{x}(t)$ мы получаем из наблюдений вида

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}).$$

Закон управления $\boldsymbol{u}(t)$ определяется с помощью процедуры минимизации функционала вида

$$Q(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}, t) = \min.$$

Этот функционал и определяет цель управления.

Пусть $\boldsymbol{x}(t_0)$ дано. Задача формулируется так, что окрестность $\boldsymbol{x}(t_1)$ достигается за некоторый фиксированный отрезок времени $T=t_1-t_0$. Тогда задачу относят к типу задач *с фиксированным временем и свободным концом траектории.*

Существует другая постановка задачи. Конец траектории строго фиксирован, то есть $\boldsymbol{x}(t_1)$ задано. Требуется найти такое управление $\boldsymbol{u}$, которое сообщает динамическому объекту траекторию, минимизирующую функционал Q. Время перехода от начального состояния к конечному не фиксировано. Тогда это задача *с закрепленными концами фазовой траектории и свободным временем*. В частном случае, взяв в качестве функционала время T, получим задачу на *максимальное быстродействие.*

Например, задача управления состоит в том, чтобы перевести космический аппарат с одной круговой орбиты на другую, тоже круговую, но более высокую. Такой перевод может быть осуществлен с помощью двух импульсов

управления. Если высота новой орбиты задана, а время такого перевода не фиксировано, то имеем задачу со свободным временем и закрепленными концами.

Второй вариант – запуск искусственного спутника Луны. С круговой орбиты вокруг Земли космический аппарат с помощью импульса переводится на орбиту, вытянутую в сторону Луны. По достижении космическим аппаратом окрестности Луны необходимо скорректировать орбиту и превратить ее в круговую около Луны. Эту коррекцию можно выполнить различными способами. Возникает задача: как сэкономить топливо? Здесь концы траектории не закреплены, а ведется поиск закона управления с минимальной энергией, решающего задачу достижения результата за ограниченное допустимое время.

Минимизация функционала – классическая задача вариационного исчисления [7]. В разных прикладных задачах те или иные преимущества имеют:

а) метод динамического программирования Р. Беллмана,

б) метод, основанный на “принципе максимума” Л. Понтрягина.

Последний – более распространен в небесной механике и астродинамике.

Принцип максимума

Запишем систему дифференциальных уравнений – математическую модель управляемого динамического объекта [8] (ограничимся автономными системами)

$$\frac{dx_j}{dt} = f_j(x_1,\ ...,\ x_n;\ u_1,\ ...\ u_l),\ \ u_r \in \Omega,\ \ j = 1,n,\ \ r = 1,l.$$

Компоненты вектора управления u_r нужно выбрать так, чтобы минимизировать функционал Q, который запишем в виде

$$Q = \int_{t_0}^{t_1} f_0(\boldsymbol{x}(\xi),\ \boldsymbol{u}(\xi)d\xi. \qquad (3.2)$$

Введем $n+1$-ю компоненту вектора состояния x

$$x_0(t) = \int_{t_0}^{t} f_0(\boldsymbol{x}(\xi),\ \boldsymbol{u}(\xi))d\xi, \qquad (3.3)$$

так, чтобы

$$\frac{dx_0(t)}{dt} = f_0(x_1,\ ...\ x_n;\ u_1,\ ...\ u_l).$$

Очевидно, что начальные условия для компоненты x_0 имеют вид $x_0(t_0) = 0$, а цель управления - $x_0(t_1) = Q$. Теперь расширенная система уравнений имеет вид

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x},\ \boldsymbol{u}),$$

где

$$\boldsymbol{x} = (x_0,\ x_1,\ ...\ x_n)^T, \boldsymbol{u} = (u_1,\ u_2,\ ...\ u_l)^T$$

Пусть $\boldsymbol{u}^0$ и $\boldsymbol{x}^0$ – оптимальное управление и оптимальная траектория соответственно. Тогда для любого процесса, мало отличающегося от оптимального, можно записать уравнение в вариациях

$$\delta \dot{\boldsymbol{x}} = \frac{\partial f(\boldsymbol{x}^0,\, \boldsymbol{u}^0)}{\partial \boldsymbol{x}} \delta \boldsymbol{x}$$

и сопряженное ему уравнение

$$\dot{\boldsymbol{\psi}} = -\left(\frac{\partial \boldsymbol{f}(\boldsymbol{x}^0,\, \boldsymbol{u}^0)}{\partial \boldsymbol{x}} \right)^T \boldsymbol{\psi}, \quad \boldsymbol{\psi} = (\psi_0, \psi_1, ..., \psi_n)^T.$$

Заметим, что $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ – квадратная матрица. Определим функцию Понтрягина (расширенную функцию Гамильтона)

$$\tilde{H}(\boldsymbol{\psi},\, \boldsymbol{x},\, \boldsymbol{u}) = \boldsymbol{\psi}^T \boldsymbol{f}(\boldsymbol{x},\, \boldsymbol{u}). \tag{3.4}$$

Теперь нашу систему можно записать в канонической форме

$$\begin{aligned} &\frac{dx_j}{dt} = \frac{\partial \tilde{H}}{\partial \psi_j}, \quad j = 0,\, 1,\, ...\, n; \\ &\frac{\partial \psi_j}{\partial t} = -\frac{\partial \tilde{H}}{\partial x_j}, \end{aligned} \tag{3.5}$$

так как

$$\frac{\partial \tilde{H}}{\partial \psi_j} = \frac{\partial}{\partial \psi_j} \left(\sum_{k=0}^{n} \psi_k f_k \right) = f_j(\boldsymbol{x},\, \boldsymbol{u}),$$

$$\frac{\partial \tilde{H}}{\partial x_j} = \sum_{0}^{n} \psi_k \frac{\partial \psi_k}{\partial x_k} = -\frac{\partial f_j}{\partial t}.$$

Л.С. Понтрягин доказал, что оптимальное управление $\boldsymbol{u} = \boldsymbol{u}^0$ достигается при максимуме функции $\tilde{H}(\boldsymbol{\psi},\, \boldsymbol{x},\, \boldsymbol{u})$

$$\tilde{M}(\boldsymbol{\psi}, \boldsymbol{x}^0) = \sup_{\boldsymbol{u} \in \Omega} \tilde{H}(\boldsymbol{\psi},\, \boldsymbol{x},\, \boldsymbol{u}). \tag{3.6}$$

Сформулируем теперь принцип максимума [8]. Пусть $\boldsymbol{u}(t)$ $(t_0 \le t \le t)$ – такое допустимое управление, что соответствующая ему траектория $\boldsymbol{x}(t)$, исходящая в момент t_0 из точки $x(t_0)$, проходит в момент t_1 через точку $x(t_1)$. Для оптимальности управления $\boldsymbol{u}(t)$ и траектории $\boldsymbol{x}(t)$ необходимо существование такой ненулевой непрерывной вектор-функции $\boldsymbol{\psi}(t)$, что

1^0 – для любого момента $t \in [t_0,\, t_1]$, являющегося точкой непрерывности управления $\boldsymbol{u}(t)$, если канонические уравнения удовлетворены, функция $\tilde{H}(\boldsymbol{\psi}, \boldsymbol{x}, \boldsymbol{u})$ переменной $\boldsymbol{u} \in \Omega$ достигает в точке $\boldsymbol{u} = \boldsymbol{u}^0$ максимума;

2^0 – в конечный момент t_1 выполнены соотношения

$$\tilde{M}(\boldsymbol{\psi}(t_1), \boldsymbol{x}(t_1)) = 0, \quad \psi_0(t_1) \le 0.$$

3.2 Решение задачи управления

Постановка задачи и решение в общем виде

Решим задачу наибыстрейшего перевода фазовой точки из одного состояния в другое. Сформулируем эту задачу следующим образом.

Пусть точка движется вдоль оси *Ox* по закону $\ddot{x}(t) = u(t), (t \geq t_0)$. Требуется найти кусочно-непрерывное управление $u(t)$, $|u(t)| \leq u^{\max}, (t_0 \leq t \leq T)$, такое, чтобы точка, выйдя из начального положения $x(t_0) = x_0, \dot{x}(t_0) = \dot{x}_0$, пришла за минимальное время в точку $x(T) = x_T, \dot{x}(T) = \dot{x}_T$ [9].

Положим, что $x_1 = x$, $x_2 = \dot{x}-$ фазовые координаты точки. Тогда задачу можно переформулировать так: быстрейшим образом перевести фазовую точку (x_1, x_2) из состояния $(x_0, \dot{x}_0)$ в состояние $(x_T, \dot{x}_T)$, считая, что движение подчиняется уравнениям

$$\dot{x}_1(t) = x_2(t), x_2(t) = u(t). \qquad (3.7)$$

Так как мы минимизируем время, то функционал Q запишется в следующем виде:

$$Q = -T = \int_0^T (-1)dt.$$

Согласно формуле (3.2) в нашем случае

$$f_0 = -1$$

и согласно формуле (3.3)

$$x_0(t) = \int_{t_0}^{t} f_0(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi))d\xi = \int_{t_0}^{t} (-1)d\xi.$$

Запишем функцию Понтрягина (3.4) в виде

$$\tilde{H}(\boldsymbol{\psi}, \boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{\psi}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = -\psi_0 + \psi_1 x_2 + \psi_2 u. \qquad (3.8)$$

Найдём сопряжённую систему из уравнений (3.5)

$$\dot{\psi}_0 = -\frac{\partial H}{\partial x_0} = 0, \dot{\psi}_1 = -\frac{\partial H}{\partial x_1} = 0, \dot{\psi}_2 = -\frac{\partial H}{\partial x_2} = -\psi_1.$$

Проинтегрировав ее, получаем

$$\psi_0 = C_1, \psi_1 = C_2, \psi_2 = -C_2 t + C_3.$$

Итак, согласно принципу максимума Понтрягина (3.6) оптимальным управлением, максимизирующем Гамильтониан (3.8), является управление

$$u_{opt}(t) = u^{\max} sign(\psi_2) = u^{\max} sign(-C_2 t + C_3). \qquad (3.9)$$

Таким образом, оптимальное управление является кусочно-постоянной функцией, принимающей значения $+u^{\max}$, $-u^{\max}$ и имеющей не более одной точки переключения t_1, при переходе через которую $u_{opt}(t)$ меняет знак.

Управление ориентацией. Движение по опорной траектории.

Теперь рассмотрим одномерную задачу управления ориентацией, считая, что тело движется по закону $\ddot{\varphi} = \varepsilon$, где φ – это угол поворота подвижной системы координат относительно неподвижной, ε – управляющее угловое ускорение $|\varepsilon(t)| \le \varepsilon^{\max}, (t_0 \le t \le T)$. Задача состоит в наибыстрейшем переводе точки фазового состояния из положения $(\varphi_0, \dot{\varphi}_0)$ в состояние $(\varphi_T, \dot{\varphi}_T)$ с тем условием, что точка $(\varphi_T, \dot{\varphi}_T)$ – лежит на некоторой заранее определённой кривой, которую будем называть *опорная* траектория. Положим, опорная траектория подчиняется закону

$$\varphi_{on} = \varphi_{on}^0 + \Omega(t - t_0), \dot{\varphi}_{on} = \Omega, \qquad (3.10)$$

где Ω – некоторая заданная величина угловой скорости, φ_{on}^0 – значение угла в начальный момент времени t_0.

Как мы выяснили раньше, оптимальным управлением будет управление (3.9) с одним переключением в момент времени t_1. В зависимости от начальной $(\varphi_0, \dot{\varphi}_0)$ и конечной $(\varphi_T, \dot{\varphi}_T)$ точек возможны два варианта управления: сначала прилагаем ускорение $-\varepsilon$, и в момент времени t_1 переключаем его на $+\varepsilon$, и наоборот. Рассмотрим оба варианта подробнее.

Управление – +

Проинтегрируем уравнение (4.1) с начальными условиями $\left(\varphi(t_0), \dot{\varphi}(t_0)\right) = (\varphi_0, \dot{\varphi}_0)$ до момента переключения t_1, приняв $u(t) = -\varepsilon$,

$$\begin{cases} \varphi(t) = \varphi_0 + \dot{\varphi}_0(t - t_0) - \dfrac{\varepsilon(t - t_0)^2}{2}, \\ \dot{\varphi}(t) = \dot{\varphi}_0 - \varepsilon(t - t_0). \end{cases} \qquad t_0 \le t \le t_1.$$

Далее произведём интегрирование этого же уравнения (4.1), но с начальными условиями $\left(\varphi(t_1), \dot{\varphi}(t_1)\right) = (\varphi_1, \dot{\varphi}_1)$ для момента переключения, приняв $u(t) = +\varepsilon$,

$$\begin{cases} \varphi(t) = \dfrac{\varepsilon t^2}{2} + (\dot{\varphi}_0 - 2\varepsilon t_1 + \varepsilon t_0)t + \varphi_0 - \dot{\varphi}_0 t_0 - \dfrac{\varepsilon t_0^2}{2} + \varepsilon t_1^2, \\ \dot{\varphi}(t) = \varepsilon t + \dot{\varphi}_0 + \varepsilon t_0 - 2\varepsilon t_1. \end{cases} \qquad t_1 \le t \le T.$$

Но при условии, что точка $(\varphi_T, \dot{\varphi}_T)$ должна лежать на опорной траектории, запишем

$$\begin{cases} \varphi_{on}^0 + \Omega(T - t_0) = \dfrac{\varepsilon T^2}{2} + (\dot{\varphi}_0 - 2\varepsilon t_1 + \varepsilon t_0)T + \varphi_0 - \dot{\varphi}_0 t_0 - \dfrac{\varepsilon t_0^2}{2} + \varepsilon t_1^2, \\ \Omega = \varepsilon T + \dot{\varphi}_0 + \varepsilon t_0 - 2\varepsilon t_1. \end{cases}$$

Таким образом, имеем систему двух уравнений с двумя неизвестными t_1 и T. При выборе корней уравнения будем руководствоваться ограничениями: $0 \le t_0 < t_1 < T$.

Управление + –

Проделаем то же самое для другого варианта управления, когда сначала включаем положительное значение управления, а в момент времени t_1 переключаемся на отрицательное

$$\begin{cases} \varphi(t) = \varphi_0 + \dot{\varphi}_0(t - t_0) + \dfrac{\varepsilon(t - t_0)^2}{2}, \\ \dot{\varphi}(t) = \dot{\varphi}_0 + \varepsilon(t - t_0). \end{cases} \qquad t_0 \le t \le t_1.$$

$$\begin{cases} \varphi(t) = -\dfrac{\varepsilon t^2}{2} + (\dot{\varphi}_0 + 2\varepsilon t_1 - \varepsilon t_0)t + \varphi_0 - \dot{\varphi}_0 t_0 + \dfrac{\varepsilon t_0^2}{2} - \varepsilon t_1^2, \\ \dot{\varphi}(t) = -\varepsilon t + \dot{\varphi}_0 - \varepsilon t_0 + 2\varepsilon t_1. \end{cases} \qquad t_1 \le t \le T.$$

Условие того, что точка $(\varphi_T, \dot{\varphi}_T)$ лежит на опорной траектории

$$\begin{cases} \varphi_{on}^0 + \Omega(T - t_0) = -\dfrac{\varepsilon T^2}{2} + (\dot{\varphi}_0 + 2\varepsilon t_1 - \varepsilon t_0)T + \varphi_0 - \dot{\varphi}_0 t_0 + \dfrac{\varepsilon t_0^2}{2} - \varepsilon t_1^2, \\ \Omega = -\varepsilon T + \dot{\varphi}_0 - \varepsilon t_0 + 2\varepsilon t_1. \end{cases}$$

На корни системы уравнений накладываются ограничения: $0 \le t_0 < t_1 < T$.

3.3. Фильтр Калмана

Для того, чтобы построить замкнутое управление, необходим канал обратной связи, который поставляет в регулятор выходное состояние объекта. Очень часто этот канал организуется с помощью датчиков, которые измеряют какую-либо компоненту вектора состояния тела. Однако, реальные датчики выдают значение фазового состояния тела с некоторой ошибкой. Одним из распространённых способов получения оценки вектора состояния является фильтр Калмана [10].

Фильтр Калмана – последовательный рекурсивный алгоритм, использующий принятую модель динамической системы для получения оценки, которая может быть существенно скорректирована в результате анализа каждой новой выборки измерений во временно*й* последовательности. Этот алгоритм находит применение в процессе управления многими сложными динамическими системами. При управлении динамической системой, прежде всего, необходимо полностью знать ее фазовое состояние в каждый момент времени. Но измерение всех переменных, которыми необходимо управлять, не всегда возможно и в этих случаях фильтр Калмана является тем средством, которое позволяет восстановить недостающую информацию посредством имеющихся неточных зашумленных измерений.

Рассмотрим модель движения $\ddot{\varphi}(t) = \varepsilon(t)$, которую можно переписать в следующем виде:

$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t)$, где $\boldsymbol{x}(t) = \begin{pmatrix} \varphi(t) \\ \dot{\varphi}(t) \end{pmatrix}$.

Таким образом,

$$\begin{pmatrix}\dot{\varphi}(t)\\ \ddot{\varphi}(t)\end{pmatrix}=\begin{pmatrix}0 & 1\\ 0 & 0\end{pmatrix}\begin{pmatrix}\varphi(t)\\ \dot{\varphi}(t)\end{pmatrix}+\begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}0\\ \varepsilon\end{pmatrix}. \qquad (3.11)$$

Модель измерений при наличии датчика угловой скорости выглядит следующим образом (записана цепочка выражений с учетом подстановок):

$$\boldsymbol{z}_k=\dot{\varphi}_k+\boldsymbol{v}_k=\boldsymbol{H}_k\boldsymbol{x}_k+\boldsymbol{v}_k=\begin{pmatrix}0 & 1\end{pmatrix}\begin{pmatrix}\varphi_k\\ \dot{\varphi}_k\end{pmatrix}+\boldsymbol{v}_k. \qquad (3.12)$$

Здесь $\boldsymbol{z}_k$ – значение непосредственного измерения в момент времени t_k, $\boldsymbol{v}_k$ – шум измерения, который обладает нулевым математическим ожиданием $E<\boldsymbol{v}(t)>=0$, то есть является Гауссовским случайным процессом. Другими словами предполагаем наличие в измерениях *белого шума*. Второй момент шума равен

$$E<\boldsymbol{v}(t)\boldsymbol{v}^T(s)>=\delta(t-s)\boldsymbol{R}(t).$$

Проинтегрируем уравнение (3.11) с начальными условиями $\left(\varphi(t_0),\dot{\varphi}(t_0)\right)=(\varphi_0,\dot{\varphi}_0)$

$$\begin{pmatrix}\varphi(t)\\ \dot{\varphi}(t)\end{pmatrix}=\begin{pmatrix}\varepsilon t^2/2+C_1t+C_2\\ \varepsilon t+C_1\end{pmatrix},\qquad \begin{pmatrix}C_1\\ C_2\end{pmatrix}=\begin{pmatrix}\dot{\varphi}_0-\varepsilon t_0\\ \varphi_0-\varphi_0t_0+\varepsilon t_0^2/2\end{pmatrix}.$$

Второй момент случайного процесса может быть описан в терминах ковариационной матрицы

$$\boldsymbol{P}(t)=E\left\langle\left[\boldsymbol{x}(t)-\hat{\boldsymbol{x}}(t)\right]\left[\boldsymbol{x}(t)-\hat{\boldsymbol{x}}(t)\right]^T\right\rangle,$$

где $\hat{\boldsymbol{x}}(t)$ – оценка состояния, а матрица $\boldsymbol{P}(t)$ называется ковариационной матрицей ошибки оценки вектора состояния. Для того чтобы получить прогнозируемую оценку ковариационной матрицы ошибки, воспользуемся матричным уравнением Риккати

$$\dot{\boldsymbol{P}}(t)=\boldsymbol{F}(t)\boldsymbol{P}(t)+\boldsymbol{P}(t)\boldsymbol{F}^T(t). \qquad (3.13)$$

Пусть ковариационная матрица ошибки $\boldsymbol{P}(t)$ имеет следующие элементы:

$$\boldsymbol{P}(t)=\begin{pmatrix}a(t) & b(t)\\ b(t) & c(t)\end{pmatrix}.$$

Тогда матричное уравнение Риккати (3.13) с начальными условиями $\begin{pmatrix}a(t_0)\\ b(t_0)\\ c(t_0)\end{pmatrix}=\begin{pmatrix}a_0\\ b_0\\ c_0\end{pmatrix}$ имеет следующее решение:

$$\begin{pmatrix}a\\ b\\ c\end{pmatrix}=\begin{pmatrix}C_1t^2+2C_2t+C_3\\ C_1t+C_2\\ C_1\end{pmatrix}\qquad \begin{pmatrix}C_1\\ C_2\\ C_3\end{pmatrix}=\begin{pmatrix}c_0\\ b_0-c_0t_0\\ a_0-2b_0t_0+c_0{t_0}^2\end{pmatrix}.$$

Таким образом, проинтегрировав модельное уравнение (3.11) и уравнение Риккати (5.2), получили априори оценки для состояния тела $\hat{\boldsymbol{x}}(-)$ и для матрицы ошибки $\hat{\boldsymbol{P}}(-)$.

Апостериори оценка $\hat{\boldsymbol{x}}_k(+)$ базируется на наблюдениях (или измерениях) $\boldsymbol{z}_k$. Таким образом, она является функцией априори оценки $\hat{\boldsymbol{x}}_k(-)$ и измерений $\boldsymbol{z}_k$ и может быть записана в следующем виде:

$$\hat{\boldsymbol{x}}_k(+) = \boldsymbol{K}_k^1 \hat{\boldsymbol{x}}_k(-) + \overline{\boldsymbol{K}}_k \boldsymbol{z}_k. \qquad (3.14)$$

Апостериори значение матрицы ошибки рассчитывается так:

$$\boldsymbol{P}_k(+) = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k)\boldsymbol{P}_k(-). \qquad (3.15)$$

Здесь $\overline{\boldsymbol{K}}_k = \boldsymbol{P}_k(-)\boldsymbol{H}_k{}^T \left[\boldsymbol{H}_k \boldsymbol{P}_k(-) \boldsymbol{H}_k{}^T + \boldsymbol{R}_k \right]^{-1}$ – так называемая матрица коэффициентов обратной связи. В нашем случае она равна

$$\boldsymbol{K}_k = \begin{pmatrix} \dfrac{b(-)}{c(-)+\boldsymbol{R}} \\ \dfrac{c(-)}{c(-)+\boldsymbol{R}} \end{pmatrix}.$$

Тогда апостериори оценка вектора состояния и матрицы ковариации из (5.4) и (5.5) соответственно, имеют вид

$$\hat{\boldsymbol{x}}_k(+) = \begin{pmatrix} \hat{\varphi}(-) + \dfrac{b(-)(z_k - \hat{\dot{\varphi}}(-))}{c(-)+\boldsymbol{R}} \\ \hat{\dot{\varphi}}(-) + \dfrac{c(-)(z_k - \hat{\dot{\varphi}}(-))}{c(-)+\boldsymbol{R}} \end{pmatrix},$$

$$P_k(+) = \begin{pmatrix} a(-) - \dfrac{b^2(-)}{c(-)+\boldsymbol{R}} & \dfrac{b(-)}{c(-)+\boldsymbol{R}} \\ \dfrac{b(-)}{c(-)+\boldsymbol{R}} & \dfrac{c(-)}{c(-)+\boldsymbol{R}} \end{pmatrix}.$$

Таким образом, получены все формулы, необходимые для работы фильтра Калмана.

3.4. Реализация алгоритма оптимального управления ориентацией

Рассмотрим некоторые особенности реализации на макете оптимального управления ориентацией, о котором шла речь в главе 3.2.

Так как двигатели по мере работы расходуют некоторое количество сжатого воздуха, масса баллонов постепенно уменьшается. Таким образом, меняется и масса самого макета, а значит и момент инерции относительно вертикальной оси также уменьшается. Тем не менее, так как давление в резервуаре макета с помощью специальных клапанов поддерживается постоянным, то удельный импульс каждого двигателя практически не меняется. Из всего сказанного следует, что ускорение, приобретаемое макетом от импульсных двигателей, изме-

няется со временем, а именно – растёт до момента, когда газ в баллонах практически закончился и его уже не хватает для поддержания постоянного давления в резервуаре. Примерная зависимость углового управляющего ускорения от времени изображена на рис.3.1. Так как ускорение зависит от времени, алгоритм управления ориентацией с некоторого времени начнёт давать ошибочные расчеты, потому что он использует некоторое постоянное значение ресурса управления. Выходом из этой проблемы является оценивание значения углового ускорения с помощью датчика угловой скорости во время каждого сеанса управления (рис.3.2). Таким образом, будем иметь для расчета времени управления каждый раз текущее значение ускорения.

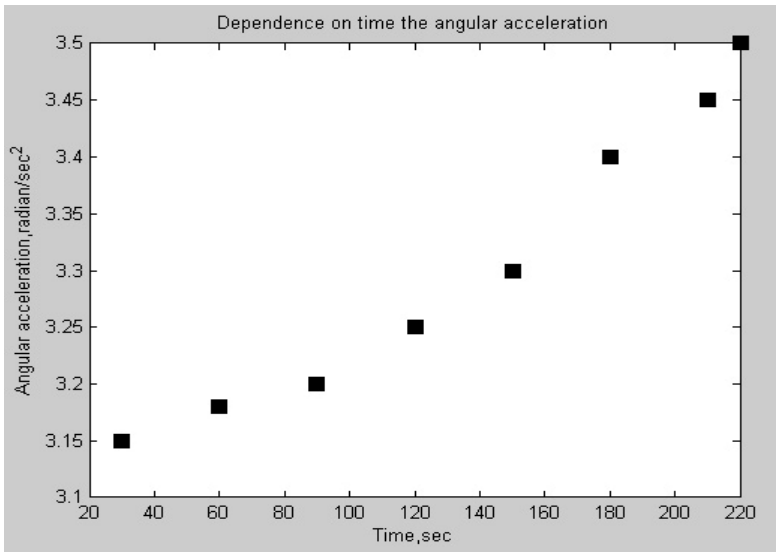

Рис.3.1. Зависимость ускорения от времени

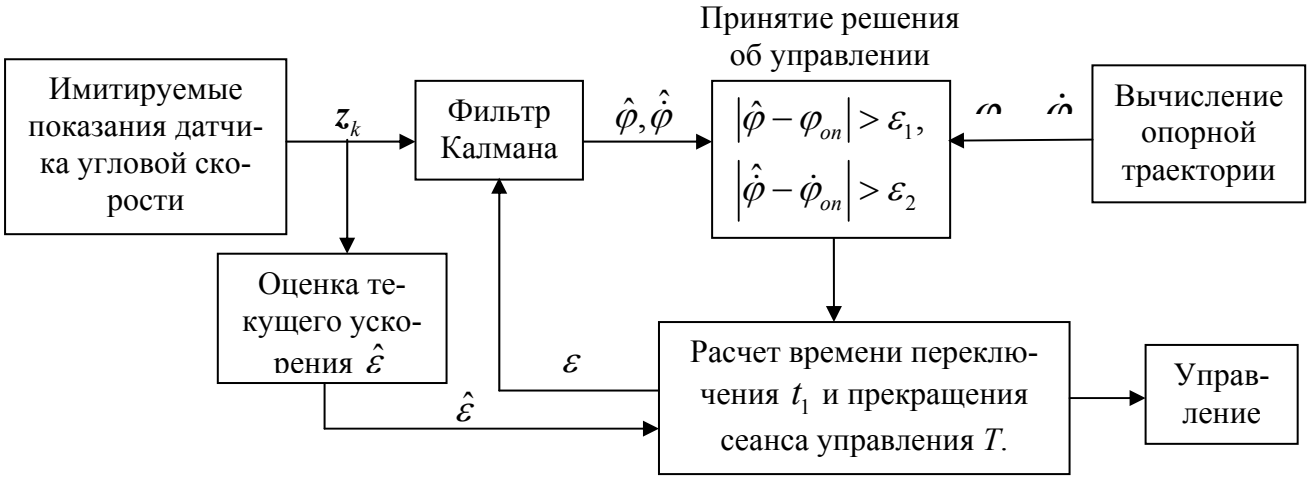

Рис.3.2. Алгоритм управления ориентацией

Другая особенность проведения экспериментов в том, что на момент их проведения имитатор звёздного неба не был полностью готов и гироскоп не был установлен на макет, поэтому управление строится только на оценке вектора состояния по имитируемым показаниям датчика угловой скорости без уточнения с помощью веб-камеры.

Нужно назвать ещё один источник ошибки в управлении, который происходит из особенности задания управления. Как было сказано выше, каждые 100 мс компьютер посылает значения времени, сколько должен работать каждый двигатель следующие 100 мс. Поэтому, когда приходит время переключить управление на противоположное, к примеру, с плюса на минус, нельзя сделать это немедленно. Сначала компьютер должен послать работающим двигателям

информацию сколько времени они должны работать в последние 100 мс, после этого времени несколько миллисекунд управление остаётся отключенным, и только в следующие 100 мс включается противоположное управление. Из-за этой особенности получается некоторая ошибка в приведении состояния тела из начального к требуемому. На общем фоне помех и трения ножек макета о стол эта ошибка не является значительной.

Эксперимент управления ориентацией проходил следующим образом. Сначала к бортовому компьютеру макета подключались клавиатура и монитор. В программе задавалась некоторая задержка начала вычислений. Далее запускалась программа, во время задержки от бортового компьютера отсоединялись монитор и мышь и макет ставился на стеклянный стол. Следующую минуту программа функционировала, решая задачу управления ориентацией каждую миллисекунду. При этом все данные с имитируемого сенсора, выходные данные из фильтра Калмана и вычисляемая опорная траектория вместе с текущим временем, которое отчитывалось с начала работы программы, – все данные сохранялись в файле.

3.5. Результаты экспериментов

В качестве опорной траектории одного из эксперимента взято движение с постоянной угловой скоростью 3 градуса в секунду. Результаты эксперимента представлены на рис. 3.3, где сплошной линией обозначена опорная траектория, а пунктирной линией – реальная траектория тела. Первый график – график скорости вращения, второй график – график ориентации, т.е. угла отклонения от некоторого начального значения, третий график – график зависимости управления от времени.

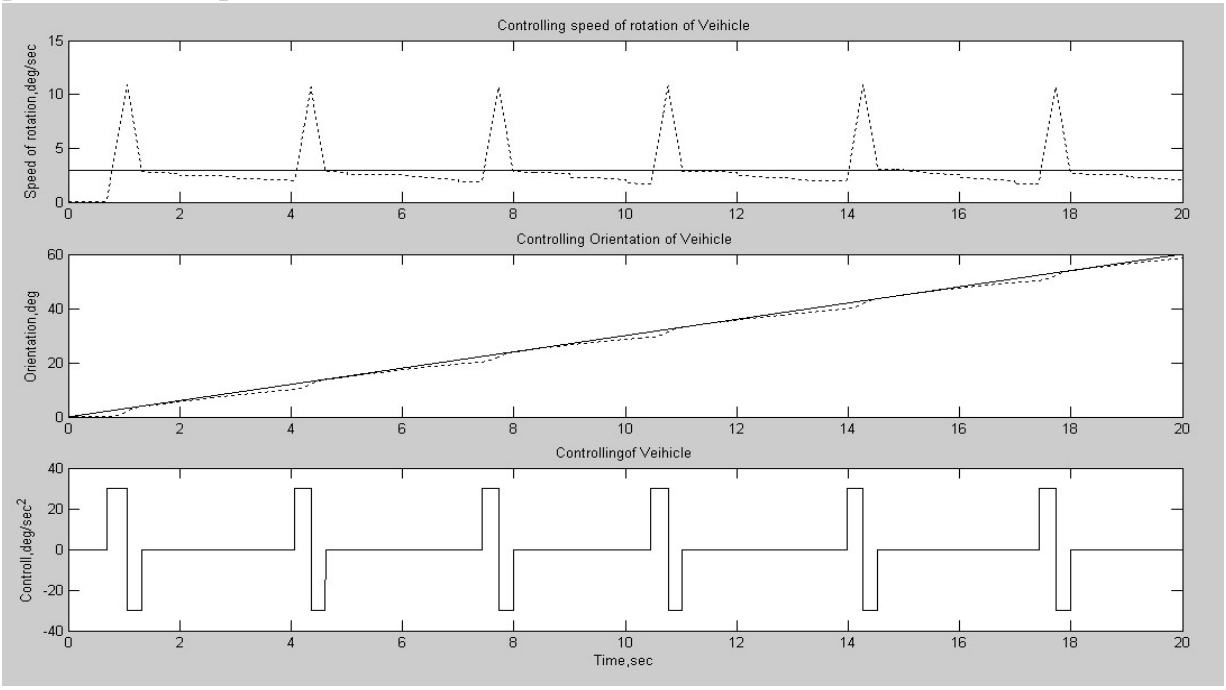

Рис.3.3. Результаты эксперимента

Вследствие неполной сборки макета, на который ещё не были установлены средства определения ориентации (гироскоп и веб-камера), управление произ-

водилось по принципу, сходному принципу разомкнутого управления. Были измерены управляющие моменты системы, возможные возмущающие факторы, и зашумленные данные датчика имитировались в программе, производящей управление макетом.

Как видно из рисунка, первый сеанс управления был вызван тем, что начальное значение угловой скорости отличалось от заданного опорного. После окончания сеанса управления траектория тела была в допустимой от опорной области. Однако, как следует из первого графика, угловая скорость тела постепенно падала вследствие небольшого трения ножек макета о стол, которое было учтено при имитации показания датчика угловой скорости. После того, как траектория тела вышла из области допустимого, включилось управление и вернуло траекторию на опорную. Далее картина повторяется, управление время от времени вмешивается в движение макета, возвращая макет на опорную траекторию.

Таким образом, можно было наблюдать, как макет первое время движется с некоторой постоянной скоростью вращения. Однако вследствие того, что управление разомкнутое, через некоторое время макет начинал менять скорость вращения, что связано с непостоянным коэффициентом трения на столе и с некоторыми случайными факторами, которые невозможно учесть в модели движения. Но, когда на макет будет установлен датчик угловой скорости, не нужно будет учитывать в модели движения возмущения, послужившие отклонению от опорной траектории. Тогда будет реализован принцип замкнутого управления, который функционирует по отклонению и которому «не важна» природа этого отклонения, будь она случайным воздействием или некоторой постоянной силой.

4. Заключение

Настоящая работа подтверждает работоспособность *нового* макета LuVeX, на котором, несмотря на неполную сборку и отсутствие важных частей (датчика угловой скорости, веб-камеры), был реализован алгоритм управления. При отработке на макете алгоритма управления ориентацией выяснилось следующее:

- рассмотренный алгоритм обеспечивает движение тела по определённой опорной траектории с некоторой наперёд заданной точностью,
- в случае, если опорная траектория построена на основе уравнений движения тела, алгоритм управления производит функцию коррекции при отклонении траектории тела от опорной,
- точность движения тела по опорной траектории имеет некоторый предел, обусловленный уровнем шума датчика и дискретностью управляющего импульса двигателей.

Таким образом, алгоритм управления импульсными двигателями, основываясь на зашумлённых измерениях датчика угловой скорости и акселерометров, обеспечивает достаточно точное нахождение тела на опорной траектории даже в случае некоторых внешних возмущений, действующих на тело.

5. Благодарности

Работа выполнена при частичной поддержке DAAD (программа Leonhard Euler, реферат 325) и РФФИ (грант 06-01-00389).

ЛИТЕРАТУРА

1. *http://ssl.mit.edu/spheres/motivation.html*
2. *http://www.afrl.af.mil/*
3. *http://sun-valley.stanford.edu/arl.html*
4. *http://www.aero.polimi.it/~space/*
5. *S.Theil, R.Stanislowski, A.Schleicher, S.Scheidhauer.* Entwurf und Entwicklung eines Luftkissenfahrzeugs für einen Satellitendynamiksimulator/ Projektbericht, Universität Bremen. – Bremen, 2003. – 38 с.
6. *М.П. Туманов.* Теория управления. Теория линейных систем автоматического управления: Учебное пособие / М: МГИЭМ, 2005. – 82с.
7. *А.И. Егоров*. Основы теории управления./М.: Физматлит, 2004. 504 с.
8. *Р Габасов., Ф.М Кириллова.* Принцип максимума в теории оптимального управления/ Минск: Наука и техника, 1974. – 271с.
9. *Ф.П. Васильев.* Численные методы решения экстремальных задач./ М.: Наука, 1988. – 552с.
10. *Д.С. Иванов, М.Ю. Овчинников.* Использование одноосного гироскопа для определения ориентации макета в лабораторных условиях/ Препринт ИПМ им. М.В. Келдыша РАН. – М., 2008. – № 11. – 32 с.
(*http://www.keldysh.ru/papers/2008/prep11/ prep2008_11.html*)

Содержание

ПРИЛОЖЕНИЕ

Текст программы управления ориентацией макета

```
void main(void)
{
void randomize(void);
FILE *source,*target,*results,*test;
int c,n,i,t;
int N;
unsigned long int timer[9],calc[7]; //массив со значениями для контроллера
calc[0]=0; calc[1]=0; calc[2]=0; calc[3]=0; calc[4]=0; calc[5]=0; calc[6]=100;
char cInp;
long CurTime=0, CurTime0=0, CTime=-99, dt=1,Tn;
struct timeb T, T1;
char str[100]= {0};
float E=180, e=0.01,Enow=0, Rand, X3; //допустимые ошибки алгоритма
float p[3], pnow[3]={0},q[2]={0},phi,w,C1,C2, a[3]={0},R, z, phiz, j;//фильтр
float PHI,W, e1, e2, tx, t1, Tr1,Nr,A,B, C, x1, x2, X;//параметры оп. траект.
a[0]=0; a[1]=0;a[2]=0;p[0]=0.01;p[1]=0.01;p[2]=0.01;//уточнение ковариации
phi=0; phiz=0;
w=0; z=0; R=0.01;
N=1000;
j=1; W=30;
PHI=0;PHI=PHI*3.14/180;//задание начальных значений для оп. траект.
W=W*3.14/180; e=e*3.14/180; E=E*3.14/180;
e1=2; e2=0.5;
e1=e1*3.14/180; e2=e2*3.14/180;//задание допустимых ошибок
target=fopen("/dev/ttyS0","wb");
results=fopen("Results30.txt","w");//откр-е файлов для записи результатов
test=fopen("Test.txt","w");
ftime(&T); ftime(&T1);
CurTime=(T1.time-T.time)*1000+T1.millitm-T.millitm;//функция времени
while(CurTime<20000)
{ ftime(&T1); CurTime=(T1.time-T.time)*1000+T1.millitm-T.millitm; }
CurTime=0; ftime(&T);
while(CurTime<60000)//программа будет выполняться 60 с.
{ftime(&T1); CurTime=(T1.time-T.time)*1000+T1.millitm-T.millitm;
 while(CurTime<CurTime0+dt)//шаг вычислений
 {ftime(&T1);
 CurTime=(T1.time-T.time)*1000+T1.millitm-T.millitm;
 }
 if(CurTime>t1*1000&&t1!=0)
 { Enow=-Enow; t1=0;    }
 if(CurTime>Tr1*1000&&Tr1!=0)
```

```
{ Enow=0; Tr1=0;          }
Rand=random();//имитация показаний датчика угловой скорости
Rand=Rand/1000000000-1.1;
z=z-0.0001+Enow*(CurTime-CurTime0)/1000+e*Rand;
phiz=phiz+z*(CurTime-CurTime0)/1000;//расчет параметров фильтра
C1=w-Enow*CurTime0/1000;
C2=phi-w*CurTime0/1000+Enow*CurTime0*CurTime0/2/1000000;
phi=Enow*CurTime*CurTime/2/1000000+C1*CurTime/1000+C2;
w=Enow*CurTime/1000+C1;
pnow[0]=p[2]*CurTime*CurTime/1000000+2*(p[1]-
p[2]*CurTime0/1000)*CurTime/1000+(p[0]-
2*p[1]*CurTime0/1000+p[2]*CurTime0*CurTime0/1000000);
pnow[1]=p[2]*CurTime/1000+(p[1]-p[2]*CurTime0/1000);
pnow[2]=p[2];
phi=phi+pnow[1]*(z-w)/(pnow[2]+R); w=w+pnow[2]*(z-w)/(pnow[2]+R);
p[0]=pnow[0]-pnow[1]*pnow[1]/(pnow[2]+R);
p[1]=pnow[1]*R/(pnow[2]+R);
p[2]=pnow[2]*R/(pnow[2]+R);
if(j>=1)// алгоритм уточнения ковариационной матрицы ошибки
{       q[0]=phi-phiz; q[1]=w-z;
        a[0]=(j-1)*a[0]/j+q[0]*q[0]*100000/j;
        a[1]=(j-1)*a[1]/j+q[0]*q[1]*100000/j;
        a[2]=(j-1)*a[2]/j+q[1]*q[1]*100000/j;
        j=j+1;}
if(j-1==N)
{       p[0]=a[0]; p[1]=a[1]; p[2]=a[2];
        a[0]=0; a[1]=0; a[2]=0;   j=1;   }
PHI=PHI+W*(CurTime-CurTime0)/1000;
if(fabs(phi-PHI)>e1&&Enow==0)// принятие решения об управлении
{       Tr1=1000000; A=-E/4; B=(-W/2+w/2+E*CurTime/2/1000);
C=-w*CurTime/1000-E*CurTime*CurTime/2/1000000+pow(-W/2 + w/2+ E*
CurTime /1000/2 ,2)/E +phi-PHI+W*CurTime/1000;
x1=(-B+sqrt(B*B-4*A*C))/2/A; x2=(-B-sqrt(B*B-4*A*C))/2/A;
     if(x1!=sqrt(-1))// вычисление времени сеансов управления
{        if((x1>0)-(x2>0)*(x1<x2)*(x1>0))
         { tx=(-W+w+E*x1+E*CurTime/1000)/2/E;
         if(tx>CurTime/1000)
         { t1=tx; Tr1=x1;
           Nr=1; Enow=-E; }
         }
         if((x2>0)-(x1>0)*(x2<x1)*(x2>0))
         { tx=(-W+w+E*x2+E*CurTime/1000)/2/E;
           if(tx>CurTime/1000)
           { t1=tx; Tr1=x2; Nr=1; Enow=-E; }
```

```
            }
        }
        A=E/4; B=-W/2+w/2-E*CurTime/2/1000;
        C=phi-w*CurTime/1000+E*CurTime*CurTime/2/1000000-pow(W/2-w/2
+E*CurTime/2/1000,2)/E-PHI+W*CurTime/1000;
        x1=(-B+sqrt(B*B-4*A*C))/2/A; x2=(-B-sqrt(B*B-4*A*C))/2/A;
        if(x1!=sqrt(-1))
        {       if((x1>0)-(x2>0)*(x1<x2)*(x1>0))
                {       if(Tr1>x1)
                                { tx=(W-w+E*x1+E*CurTime/1000)/2/E;
                                        if(tx>CurTime/1000)
                                        { t1=tx; Tr1=x1; Nr=2; Enow=E;}
                                }
                }
                if((x2>0)-(x1>0)*(x2<x1)*(x2>0))
                {       if(Tr1>x2)
                                { tx=(W-w+E*x2+E*CurTime/1000)/2/E;
                                                if(tx>CurTime/1000)
                                        { t1=tx; Tr1=x2; Nr=2; Enow=E; }
                                }
                }
        }
        }
        CurTime0=CurTime;               //запись всех данных в файл
        printf("%f %f %f %f %f %f %f%d\n", z*180/3.14, phiz*180/3.14, w*180/3.14,
phi*180/3.14, W*180/3.14, PHI*180/3.14, Enow*180/3.14, CurTime);
        fprintf(results,"%f %f %f %f %f %f %f %d\n", z*180/3.14, phiz*180/3.14,
w*180/3.14, phi*180/3.14, W*180/3.14, PHI*180/3.14, Enow*180/3.14, CurTime);
        if(CurTime==CTime+100) // передача вычисленных значений контроллеру
        {       if(Enow>0)
                {       if(t1!=0)
                        {X3=t1*1000; Tn=(long)X3; }
                        else    {  X3=Tr1*1000;  Tn=(long)X3; }
                        if(Tn>CurTime+100)
                                { calc[1]=100; calc[3]=100;}
                        else    {       if(Tn-CurTime!=10)
                                        { calc[1]=Tn-CurTime; calc[3]=Tn-CurTime; }
                                        else { calc[1]=0; calc[3]=0; }
                                }
                }
                if(Enow<0)
                {       if(t1!=0)
                        { X3=t1*1000; Tn=(long)X3; }
                        else    { X3=Tr1*1000; Tn=(long)X3;}
```

```
                    if(Tn>CurTime+100)
                    { calc[0]=100; calc[4]=100; }
                    else    {       if(Tn-CurTime!=10)
                                    { calc[0]=Tn-CurTime; calc[4]=Tn-CurTime; }
                                    else    { calc[0]=0; calc[4]=0;}
                            }
                    }
            }
            if(CurTime>=CTime+100)
            { int k,l;
            for(k=1;k<8;k++)
            {timer[k]=calc[k-1];}
            timer[0]=101;               //header
      timer[8]=  (2*(timer[7]+timer[6]+timer[5]  +timer[4]+timer[3]+timer[2]+  timer[1])); //<-checksum
       for(i=0;i<9;i++)
       {cInp=(char)timer[i]; fputc(cInp,target); fflush(target); } // посылка контроллеру
       timer[8]=0;  CTime=CurTime; // запись значений управлений в файл
       fprintf(test,"%i %u %u %u %u %u %u %i\n", Tn,timer[1], timer[2], timer[3], timer[4], timer[5], timer[6], timer[7], CurTime);
       for(i=0;i<6;i++)
       { timer[i+1]=0; calc[i]=0; }
       }
      }
      fclose(results);
      fclose(target);
      return 0;
      }
```