электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

ПОСТРОЕНИЕ ОТКРЫТОЙ СИСТЕМЫ ДИНАМИЧЕСКОГО АНАЛИЗА КОНЕЧНО-ЭЛЕМЕНТНОГО КОМПЛЕКСА

 # 10, октябрь 2016

Ломакин В. В.[1,*]

УДК: 519.688

[1]Россия, МГТУ им. Н.Э. Баумана
*lomakin_vv@mail.ru

Введение

На данный момент значительная часть задач расчетного анализа несущих конструкций решается с использованием метода конечных элементов [1], который получил широкое распространение и сейчас реализован во многих программных комплексах (ПК). Применение существующих ПК для решение общеинженерных задач как правило достаточно эффективно и затруднений не вызывает. Решение же специфических динамических задач вызывает определенные сложности. Поэтому у пользователя нередко возникает необходимость в создании собственного конечно-элементного ПК или решателя, ориентированного на подобные задачи.

При создании конечно-элементных ПК для расчетного анализа различных конструкций сложно заранее спрогнозировать, что потребуется в будущем при их дополнении и развитии. И если для подсистемы статического анализа это не так актуально, поскольку их компоненты (типы конечных элементов, виды узловых связей, распределение нагрузок и т.д.), как правило, статичны, то для подсистемы динамического анализа оказывается важным понимание того, как она может развиваться. Под развитием или модернизацией будем понимать в первую очередь дополнение системы расчетного анализа новыми элементами, связями, возможностями расчета, симуляциями систем управления и т.п. без перекомпоновки базовых блоков системы динамического анализа.

Для целей вмешательства и управления работой системы необходимо для каждого этапа алгоритма предусмотреть запуск дополнительных, в общем случае неизвестных на этапе создания системы анализа, модулей, подключаемых только во время работы. Для этих целей хорошо подходят динамически подключаемые библиотеки, которые в дальнейшем и будем подразумевать под программными модулями, реализующими те или иные «динамические» элементы.

Минимально необходимый набор «динамических» элементов для динамического анализа можно определить таким образом:

1. Воздействия, воспринимаемых расчетной моделью:
    - силовое, зависящее от времени;
    - кинематическое, зависящее от времени.
2. Узловые связи:
    - линейные и нелинейные упруго-демпфирующие связи с заданными характеристиками (функции усилий в зависимости от относительных смещений и скоростей узлов);
    - линейные и нелинейные упруго-демпфирующие связи с заданными характеристиками (функции усилий в зависимости от относительных смещений и скоростей узлов) с возможностью формирования гистерезисных функций (т.е. элементы «с памятью»), на самом деле не являются действительно необходимыми, однако широко используются в анализе систем амортизации с пневмогидравлическими амортизаторами.
3. Связи типа «точка – линия», «точка – поверхность» с заданными характеристиками, реализующими в том числе односторонний контакт.

Этот набор «динамических» элементов перекрывает основные потребности при исследовании динамики достаточно широкого класса агрегатов. Набор может быть встроен в подсистему динамического анализа, или выполнен в виде подключаемых библиотек.

Требования к построению открытой системы динамического анализа

Подсистему динамического анализа можно рассматривать как некий сервисный модуль, запускающий те или иные программные модули, связывающий их воедино по определенному алгоритму. Данный алгоритм, в общем случае, выглядит следующим образом (рис.1):

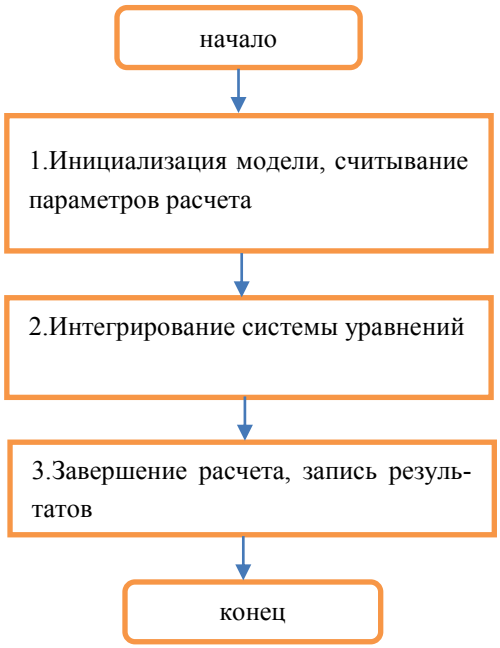

Рис.1. Алгоритм сервисного модуля

В начале и конце каждого этапа базового алгоритма необходимо вызывать процедуры программных модулей, реализующих различные «динамические» элементы, сервисные процедуры и т.д. В самом общем случае, так как «динамические» элементы могут быть зависимыми друг от друга, необходимо соблюдать порядок запуска процедур.

Исходя из выше изложенного можно сформулировать базовые требования к построению открытой системы динамического анализа:

1. Система должна запускать процедуры из программных модулей (даже неизвестных на момент ее создания) в определенном порядке, который в общем случае должен определять разработчик этих программных модулей. Порядок запуска процедур, их состав для каждого этапа можно назвать определенным «проектом» расчета, информация о котором записана в файл. Соответственно, необходимо определить единый интерфейс запускаемых процедур.
2. Процедур в программном модуле может быть несколько, более того один программный модуль (библиотека) может определять несколько «динамических» элементов, соответственно наименование процедур не должно быть чем-то ограничено.
3. Необходимо иметь возможность прервать расчет на любом шаге интегрирования.
4. Система должна предоставлять возможность использования различных методов интегрирования: как одно- так и многошаговых явных и неявных.
5. Система должна предоставлять подключаемым модулям процедуру интегрирования, если они в ней нуждаются.
6. Система должна обеспечивать возможность для любого «динамического» элемента инициировать переформирования динамической матрицы.
7. Система должна обеспечивать доступ ко всем данным и результатам расчета в момент расчета, в том числе к данным «динамических» элементов, о которых ничего неизвестно на момент ее создания.
8. Поскольку поиск необходимых данных по запросу программных модулей, как правило, требует значительного времени, то желательно осуществлять его один раз в момент инициализации и в дальнейшем не использовать на каждом шаге интегрирования.
9. Поскольку данные (и результаты) могут быть модифицированы при работе программных модулей, то они должны предоставляться в виде постоянных ссылок (или указателей) на данные.
10. Система должна предоставлять интерфейс для вывода критических сообщений пользователю от любого подключаемого программного модуля.
11. Система должна предоставлять возможность записи результатов в базу данных модели.
12. Система должна предоставлять возможность записи в лог-файл (регистрационный файл) динамического анализа модели для дальнейшего анализа.
13. Каждый программный модуль системы отвечает только за «свои» данные (или данные модуля, которые он модифицирует) и ничего не обязан «знать» про «чу-

жие» данные, не должны делаться никакие предположения о порядке следования процедур вычисления, а если о порядке вычислений модуль должен «знать», то он должен этот порядок вычислений запрашивать у системы.

Кроме этих базовых требований можно сформулировать еще несколько, касающихся непосредственно программной реализации:

1. При вызове любой процедуры программного модуля любого «динамического» элемента необходимо передавать ей минимально необходимый набор сервисных функций системы, который проще оформить в виде структуры с полями – ссылками на сервисные функции заранее заданного интерфейса.
2. Каждый «динамический» элемент может иметь как свой собственный файл данных с параметрами для конкретной модели. При этом имя модели модуль может запрашивать у системы. Кроме того, модуль при необходимости, может использовать базу данных модели. В этом случае система должна предоставлять данные «динамического» элемента по запросу через сервисную функцию.
3. Каждый «динамический» элемент обязан иметь функцию, отвечающую за поиск данных этого элемента для системы. Данные могут идентифицироваться, например, в точечной нотации: idAmor1.L[i], где «idAmor1» - имя «динамического» элемента (в данном случае - упруго-демпфирующей связи; «L» - массив длин упруго-демпфирующей связи; «i» - номер упруго-демпфирующей связи). При этом задание для поиска идентифицирующей строки «idAmor» должно приводить к выдаче массива параметров всех упруго-демпфирующих связей модели. Эта функция должна иметь заданный интерфейс и регистрироваться в системе в момент инициализации «динамического» элемента. В этом случае наименование этой функции в конкретном программном модуле не будет иметь значения.
4. Желательно (но не обязательно), чтобы подсистема динамического анализа была оформлена в виде подключаемой библиотеки. В этом случае появляется возможность программно управлять последовательностью расчетов одной и той же модели при, например, поиске оптимальных параметров.

Алгоритм работы подсистемы динамического анализа

Список событий (точек алгоритма подсистемы динамического анализа), которым должны соответствовать процедуры (функции) программных модулей «динамических» элементов, может выглядеть следующим образом:

1. OnBeforInit – событие происходит перед инициализацией модели и считыванием данных по модели, параметров расчета и т.д.
2. OnAfterInit – событие происходит после инициализации модели.
3. OnBeforSetWeight – событие происходит перед вычисление вектора веса модели.
4. OnAfterSetWeight – событие происходит перед вычисление вектора веса модели.

5. OnBeforInitElements – событие происходит перед инициализацией встроенных (стандартных) для подсистемы динамического анализа «динамических» элементов.
6. OnInitElements – инициализация встроенных (стандартных) для подсистемы динамического анализа «динамических» элементов.
7. OnAfterInitElements – событие происходит после инициализации встроенных (стандартных) для подсистемы динамического анализа «динамических» элементов.
8. OnAfterInit - событие происходит после инициализации модели и всех стандартных и подключаемых «динамических» элементов.
9. OnGetData – событие происходит при поиске данных системой. Именно здесь каждому «динамическому» элементу необходимо обеспечить поиск и выдачу своих данных по запросу системы.
10. OnBefor_BeforCalc – событие возникает перед процедурой начала динамического расчета.
11. OnBeforCalc – процедура подготовки к началу расчета. Необходима для многошаговых методов интегрирования.
12. OnAfter_BeforCalc – событие возникает после процедуры начала динамического расчета.
13. OnBeforSetDynamicMatrix – событие возникает перед началом процедуры формирования динамической матрицы.
14. OnSetDynamicMatrix – событие возникает при формировании динамической матрицы.
15. OnAfterSetDynamicMatrix – событие возникает после процедуры формирования динамической матрицы.
16. OnBeforPrivDynMatrix – событие возникает перед приведением (например, по Гауссу) динамической матрицы.
17. OnAfterPrivDynMatrix – событие возникает после приведения (например, по Гауссу) динамической матрицы.
18. OnBeforSetPower – событие возникает на каждом шаге интегрирования перед запуском процедуры формирования вектора усилий.
19. OnSetPower – событие возникает на каждом шаге интегрирования для формирования вектора усилий.
20. OnAfterSetPower – событие возникает на каждом шаге интегрирования после процедуры формирования вектора усилий.
21. OnBeforPrivPower – событие возникает на каждом шаге интегрирования перед приведением (например, по Гауссу) вектора усилий.
22. OnAfterPrivPower – событие возникает на каждом шаге интегрирования после приведения (например, по Гауссу) вектора усилий.

23. OnBeforCalcPrognoz – событие возникает на каждом шаге интегрирования перед вычислением прогнозных значений векторов перемещений, скоростей и ускорений модели.
24. OnAfterCalcPrognoz – событие возникает на каждом шаге интегрирования после вычисления прогнозных значений векторов перемещений, скоростей и ускорений модели.
25. OnBeforStep – событие возникает на каждом шаге интегрирования перед запуском процедуры шага интегрирования.
26. OnAfterStep – событие возникает на каждом шаге интегрирования после процедуры шага интегрирования.
27. OnBeforCalc_S, OnBeforCalc_V, OnBeforCalc_A – события, возникающие на каждом шаге интегрирования, перед вычислением соответсвенно: перемещений, скоростей и ускорений модели.
28. OnAfterCalc_S, OnAfterCalc_V, OnAfterCalc_A – события возникающие на каждом шаге интегрирования, после вычисления соответственно: перемещений, скоростей и ускорений модели.
29. OnBeforCalcElement – событие возникает на каждом шаге интегрирования перед вычислением усилий (или иных необходимых данных/результатов) стандартных для подсистемы динамического анализа «динамических» элементов.
30. OnAfterCalcElement – событие возникает на каждом шаге интегрирования после вычисления усилий (или иных необходимых данных/результатов) стандартных для подсистемы динамического анализа «динамических» элементов.
31. OnBeforSaveElement – событие возникает перед записью результатов для стандартных для подсистемы динамического анализа «динамических» элементов.
32. OnAfterSaveElement – событие возникает после записи результатов для стандартных для подсистемы динамического анализа «динамических» элементов.
33. OnBeforDestroyElements – событие возникает перед процедурой освобождения памяти для стандартных «динамических» элементов подсистемы.
34. OnAfterDestroyElements – событие возникает после процедуры освобождения памяти для стандартных «динамических» элементов подсистемы.
35. OnBeforFinish – событие возникает перед процедурой завершения расчета и освобождения памяти.
36. OnAfterFinish – событие возникает после процедуры завершения расчета и освобождения памяти и является последней в цепи событий.

Исходя из представленных требований и указанных обязательных событий (точек алгоритма) можно переработать алгоритм работы подсистемы динамического анализа (рис.2 – рис.6):

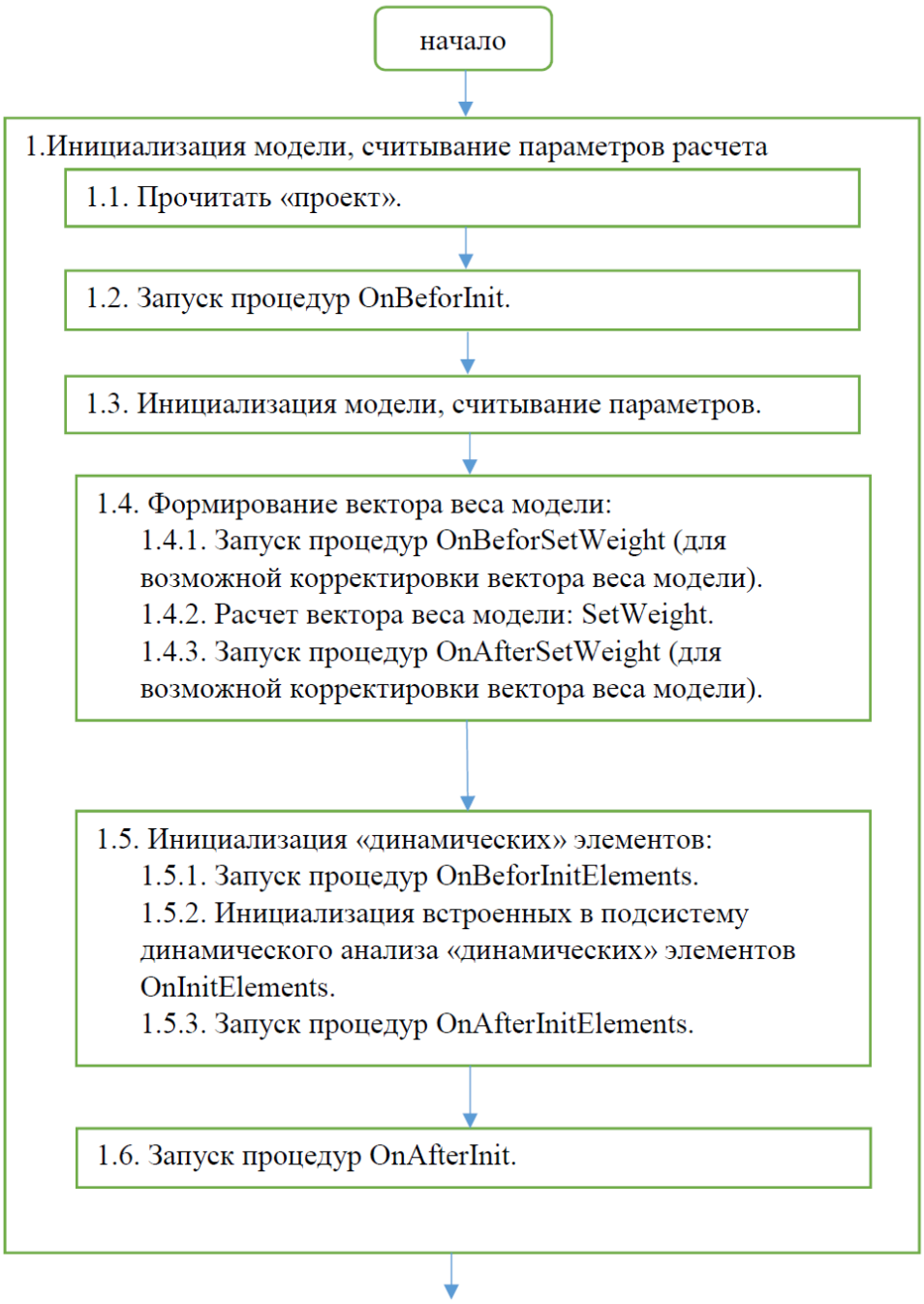

Рис. 2. Алгоритм работы подсистемы динамического анализа

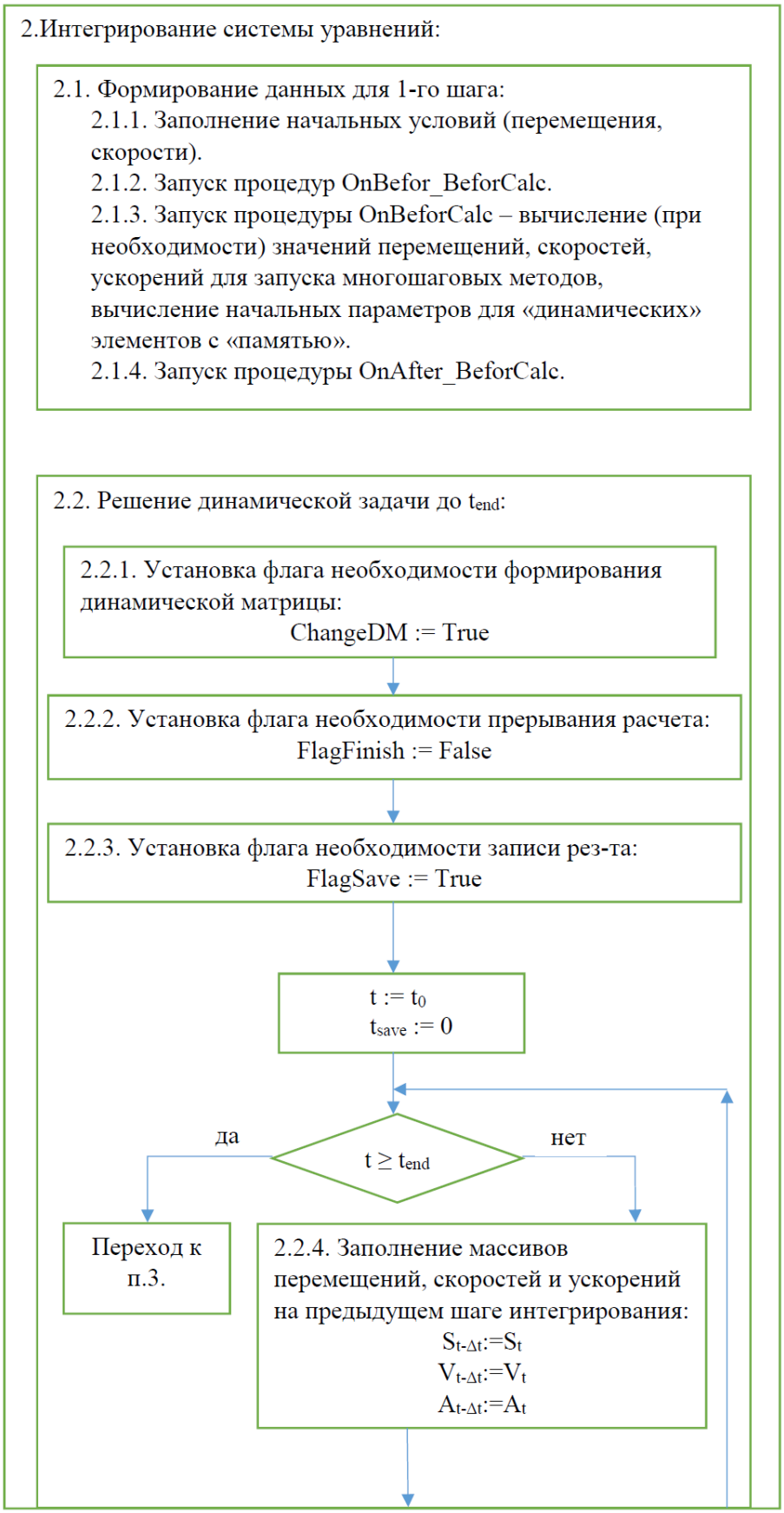

Рис. 3. Алгоритм работы подсистемы динамического анализа

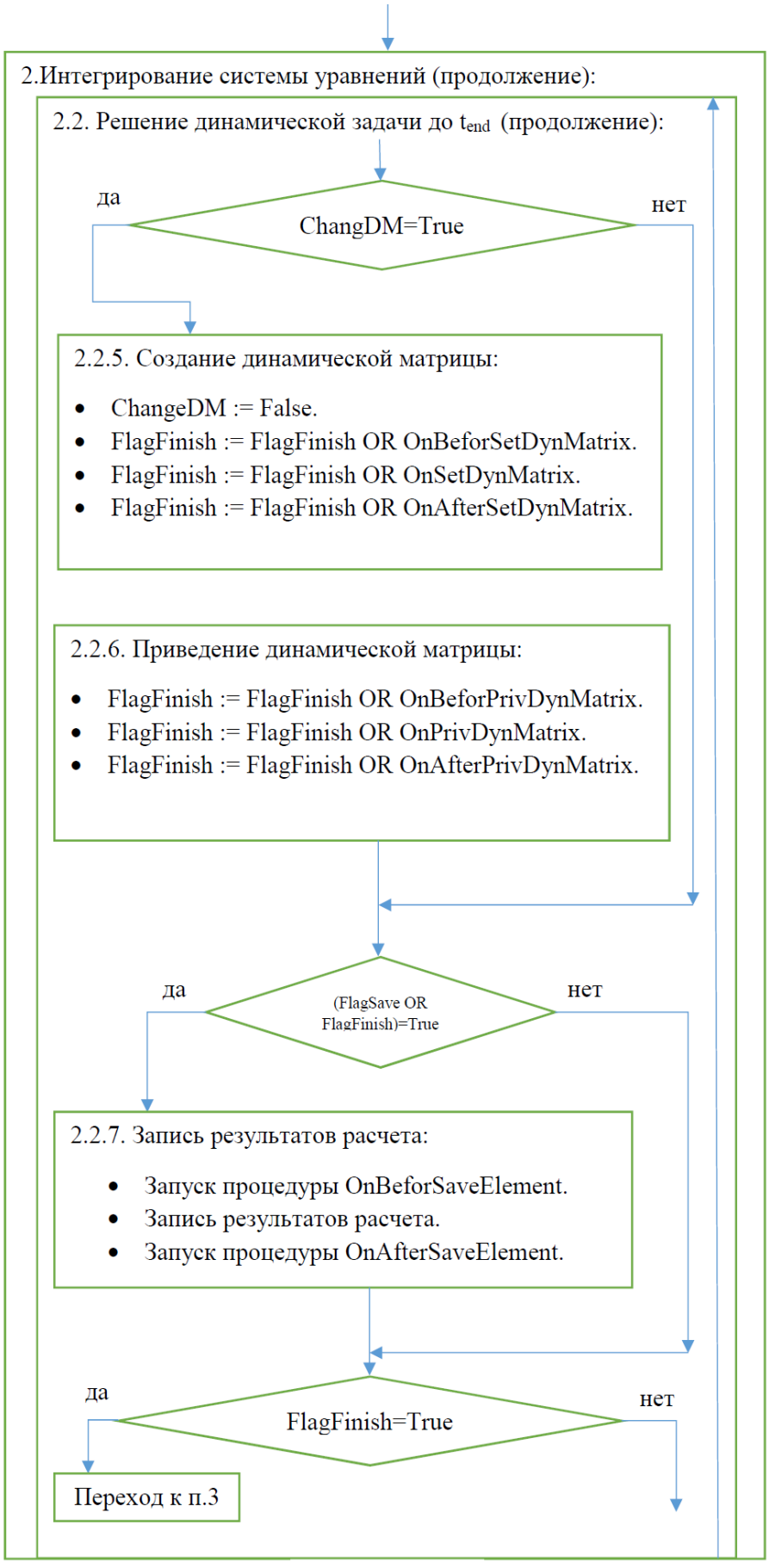

Рис. 4. Алгоритм работы подсистемы динамического анализа

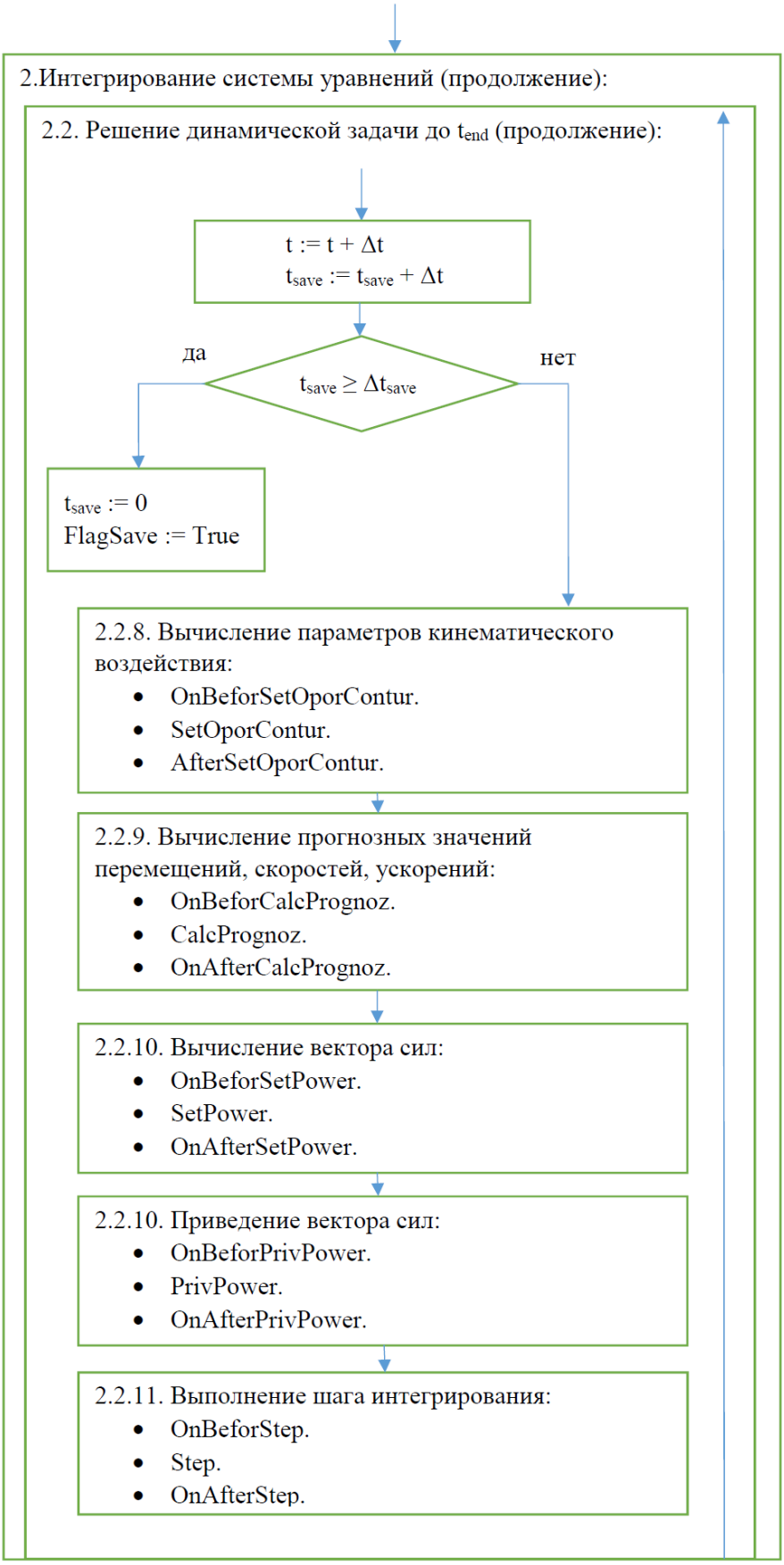

Рис. 5. Алгоритм работы подсистемы динамического анализа

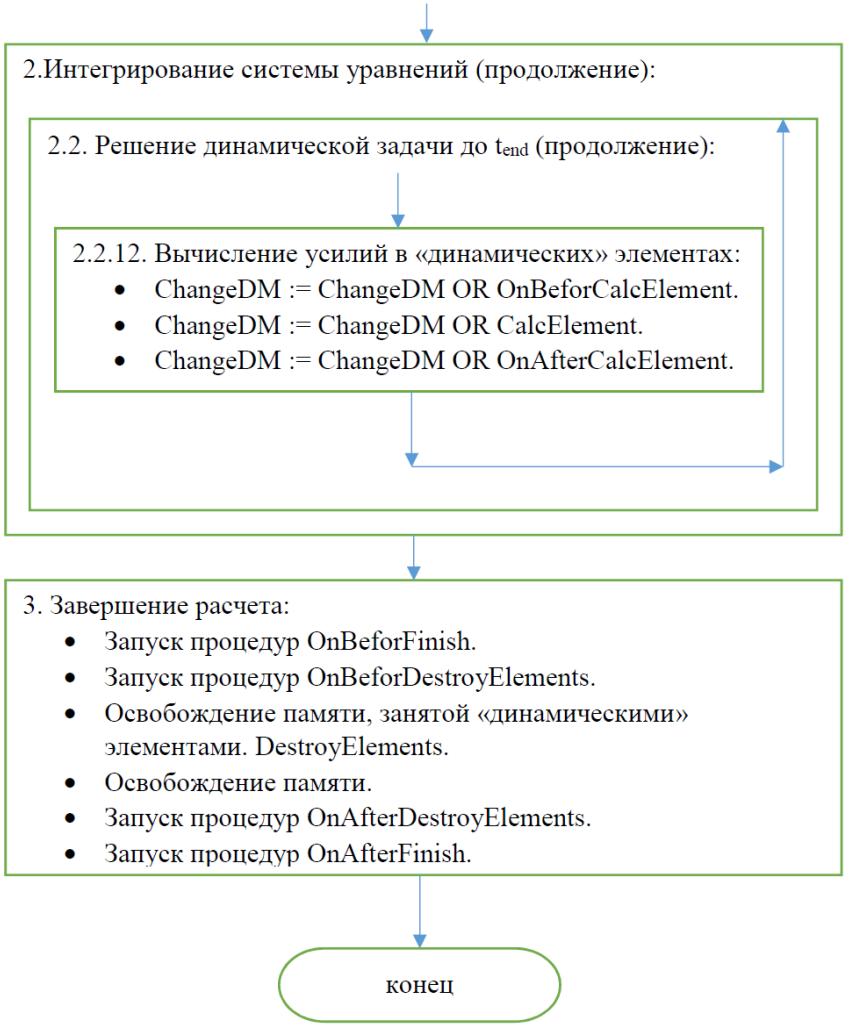

Рис. 6. Алгоритм работы подсистемы динамического анализа

Заключение

Применение вышеизложенного подхода к построению динамической системы позволяет использовать отлаженные модули без изменений, т.к. все модификации можно вынести в новые «динамические» элементы, либо использовать библиотеки, модифицирующие поведение используемых «динамических» элементов. Кроме того, возникает возможность использовать различные методы интегрирования систем уравнений без изменения базового алгоритма. При этом все остальные блоки, поскольку они независимы, остаются теми же самыми. В том числе появляется возможность использовать различные методы решения системы линейных алгебраических уравнений, поскольку они зависят от метода интегрирования.

Такой подход к проектированию подсистемы динамического анализа применен в ПК «SADAS», созданной на кафедре «Стартовые ракетные комплексы» (СМ-8) МГТУ им. Н.Э. Баумана. «Динамические» элементы новых типов были созданы и применены для решения задач, например, моделирования сферического подпятника [2], учета аэродинамики самолета в задаче выброса ракеты из самолета [3] и др.

Список литературы

[1]. Батэ К., Вилсон Е. Численные методы анализа и метод конечных элементов. М.: Изд-во «Озон». 2012. 445 с.

[2]. Ломакин В.В. Моделирование развитого сферического подпятника в элементах конструкций агрегатов космической техники. // Инженерный вестник. Электронное научно-техническое издание. МГТУ им. Н.Э. Баумана. 2014. №12. Режим доступа: http://engbul.bmstu.ru/doc/744413.html (дата обращения: 1.10.2016)

[3]. Александров А.А., Драгун Д.К., Забегаев А.И., Ломакин В.В. Механика контейнерного старта ракеты при действии поперечных нагрузок. // Инженерный журнал: Наука и инновации. Электронное научно-техническое издание. 2013. №3. DOI: 10.18698/2308-6033-2013-3-631. Режим доступа: http://engjournal.ru/catalog/machin/rocket/631.html (дата обращения: 1.10.2016)