УДК 624.042.8:629.73:531

АЛГОРИТМ РАСЧЕТА КОЛЕБАНИЙ СТЕРЖНЕВЫХ КОНСТРУКЦИЙ НА ОСНОВЕ ОБЩЕГО УРАВНЕНИЯ ДИНАМИКИ И МЕТОДА КОНЕЧНЫХ ЭЛЕМЕНТОВ

Распопов А.С.
Артемов В.Е.
Русу С.П.
Днепропетровский национальный университет железнодорожного транспорта имени академика В. Лазаряна

Область прикладной инженерии насчитывает большое количество разнообразных по назначению и потенциалу компьютерных программ – от сравнительно небольших до универсальных комплексов, предназначенных для моделирования различных физических явлений и процессов. Основными требованиями, предъявляемыми к расчетному программному обеспечению, являются надежность, универсальность, поддержка пользователей, наличие легкодоступной справочной системы, быстрота освоения программы. Однако далеко не все расчетные пакеты сегодня отвечают указанным требованиям. Быстрота расчета, надежность и универсальность присущи дорогостоящим комплексам (например, *Ansys*, *Nastran*, *Adams*), уже зарекомендовавшим себя на рынке инженерного программного обеспечения. Однако, на подготовку квалифицированного пользователя такого комплекса уходит достаточно много времени. Вместе с тем, инженер-расчетчик зачастую не нуждается во всех функциях большого программного комплекса, а использует либо профильные расчетные пакеты, направленные на решение конкретных задач, либо применяет математическое моделирование в любой доступной алгоритмической среде или среде программирования. Последний вариант даже при наличии минимума знаний по составлению алгоритмов и программированию может существенно упростить крупные блоки расчета, ускорив осуществление поставленных задач в целом.

Мостовые конструкции в преобладающем большинстве представляют собой комбинации горизонтальных, вертикальных и наклонных стержневых элементов. Для статического расчета таких систем успешно применяются классические методы строительной механики – методы сил, перемещений, смешанный, методы граничных (МГЭ) и конечных (МКЭ) элементов. Между ними и методами механики абсолютно твердого тела существуют некоторые различия. Твердотельная механика успешно разрешает некоторые уникальные задачи, актуальные для мостостроения. Это, в первую очередь, определение конечных перемещений массивных элементов конструкций, возможность учета режимов движения временной нагрузки и определение ее динамических характеристик, контактные задачи, вычисление параметров ориентации в пространстве без использования тригонометрических функций и др. Системы дифференциальных уравнений, построенные на основе общего уравнения динамики твердого тела, хорошо интегрируются известными методами и удобны для программирования на компьютере с помощью динамических массивов и матриц.

Динамическое поведение пролетного строения моста под движущейся нагрузкой характеризуется появлением существенных сил инерции. Чтобы определить кинематические параметры для всех элементов такой системы, необходимо установить законы движения центров масс этих элементов. В случае плоской системы с небольшим числом степеней свободы эта операция не представляет особого труда и решается аналитически. В сложных конструкциях применяют численное интегрирование. Как известно, наиболее общими

уравнениями движения как голономных, так и неголономных механических систем, являются уравнения движения в квазикоординатах или уравнения Эйлера–Лагранжа [1]. Так, для каждого k -го элемента динамической системы, состоящей из n элементов, можно записать

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{\pi}_k} - \frac{\partial T^*}{\partial \pi_k} + \gamma_{kij}\frac{\partial T^*}{\partial \dot{\pi}_i}\dot{\pi}_j = \Pi_k, \quad k = 1, 2, \ldots, n. \tag{1}$$

Это уравнение является следствием общего уравнения динамики системы. Его подробное аналитическое представление и координатная форма приведены в работе [2]. Отметим, что здесь величина $\Pi_k = Qb_k$ представляет собой обобщенную силу на виртуальном перемещении $\delta\pi_k$, которая при рассмотрении реальных стержневых конструкций зависит от типа связующего стержневого элемента, передающего усилие, т. е. от типа конечного элемента. Таким образом, правая часть уравнения (1) может быть представлена через главный вектор внутренних усилий (реакций) в узлах конечно-элементной сетки.

Теоретическим основам метода конечных элементов и его приложениям к расчету строительных конструкций посвящено большое количество научных трудов [3, 4]. Пример реализации метода конечных элементов в математическом пакете Mathcad можно найти в работе [5]. Язык программирования *Object Pascal* (*Delphi*) также представляет удобные средства для программирования научных и инженерных задач. Следуя основным понятиям объектно-ориентированного программирования, для описания конечно-элементной стержневой модели введем 7 классов: класс пространственных координат (координаты и углы поворота); класс кинематических закреплений с булевыми признаками (1 или 0); класс узловой нагрузки; класс стержневого элемента; класс узла конечно-элементной сетки; класс траектории движения нагрузки; класс объединяющей системы. На языке программирования *Object Pascal* эти классы могут быть представлены следующим образом.

```
TCoordinate = class
   x, y, z, rx, ry, rz :TElement;
end;


TDisplacement = class
   x, y, z, rx, ry, rz :Boolean; {степень свободы / кинематическое закрепление}
end;

TForceTorque = class
   Fx, Fy, Fz, Tx, Ty, Tz :TElement;
end;

TRod = class
   E, G, A, L, Jx, Jy, Jz :TElement; {характеристики}
   begin_node, end_node :Integer; {номера узлов начала и конца стержня}
   {Блоки матрицы жесткости для стержня}
   function C11 :TMatrix;
   function C12 :TMatrix;
   function C21 :TMatrix;
   function C22 :TMatrix;
   function C :TMatrix;  {матрица жесткости для стержня}
   {Силы реакции на концах стержня}
```

```
   function ReactionA(UA, UB :TVector) :TVector;
   function ReactionB(UA, UB :TVector) :TVector;
   {Эпюры внутренних усилий на концах стержня}
   function EpuraA(UA, UB :TVector) :TVector;
   function EpuraB(UA, UB :TVector) :TVector;
end;

TNode = class
   Coord :TCoordinate;       {координаты узлов}
   Disp :TDisplacement;      {кинематические закрепления узлов}
   FT :TForceTorque;         {вектор узловых нагрузок}
   Drift :TCoordinate;       {вектор узловых начальных смещений}
   IsDrift :TDisplacement;   {признаки заданных начальных смещений}
end;


TPath = class

   n :Integer; {количество стержней}
   function AddRod(NumRod :Integer) :Integer;
end;

TConstruction = class
   function GetRodInPath(coord :TCoordinate; NumPath :Integer) :Integer; {номер стержня, на
который действует нагрузка}
   function GetSTIFF :TMatrix; {общая матрица жесткости}
   function AddNode(Node :TNode) :Integer;
   function AddRod(Rod :TRod) :Integer;
   function AddPath(Path :TPath) :Integer;
   function AddLoad(coord :TCoordinate; F :TForceTorque; NumPath :Integer) :Integer;
   function AddLoadNode(rod :Integer; coord :TCoordinate; F :TForceTorque) :Integer;
   function NodeCoords :TVector; {координаты узлов}
   function DOF_DOF :TVector; {степени свободы}
   function DOF :TVector; {кинематические закрепления}
   function FT_DOF :TVector; {вектор узловых нагрузок по направлениям, свободным от
кинематических закреплений}
   function FT :TVector; {общий вектор узловых нагрузок}
   function Drift :TVector; {вектор начальных смещений}
   function IsDrift :TVector; {вектор признаков начальных смещений}
   function NodeSTIFF(N_Node :Integer) :TMatrix; {матрица жесткости для заданного узла}
   function STIFF_DOF :TMatrix; {общая матрица жесткости с учетом кинематических
закреплений}
   function Translation_DOF :TVector; {перемещения узлов с учетом кинематических
закреплений}
   function Translation :TVector; {перемещения узлов}
   function NodeReaction :TVector; {реакции в узлах}
   function ReactionTable :TMatrix; {реакции в стержнях}
   function DriftToFT :TVector; {эквивалентные силы}
   procedure SetNodeLoads(Loads :TVector); {начальная нагрузка}
```

В приведенных классах используются типы данных *TElement* (действительное вещественное число), *TVector* (вектор-столбец с вещественными ячейками), *TMatrix* (матрица). Класс *TCoordinate* содержит 6 вещественных чисел для описания положения и ориентации. Класс *TDisplacement* описывает кинематическое закрепление узла по булевому признаку, обеспечивая учет опор в системе. Класс *TForceTorque* содержит 6 компонент для описания узловой нагрузки по всем степеням свободы. Класс *TRod* включает информацию о жесткостных параметрах и материале стержневого элемента, а также номера узлов и функции получения матриц жесткости для концевых сечений стержня. Класс узла конструкции *TNode* содержит основные подклассы *TCoordinate*, *TDisplacement*, *TForceTorque*, а также дополнительные подклассы *TCoordinate* и *TDisplacement* для описания вынужденных смещений (кинематического возмущения). Класс *TPath* содержит массив стержней, к которым прикладывается система подвижных нагрузок, и по сути представляет собой траекторию движения нагрузки. Класс *TConstruction* объединяет всю рассчитываемую конструкцию в единый класс, содержащий массивы узлов, стержней, траекторий нагружения, а также системы уравнений равновесия, совместности деформаций и физические уравнения в линейно-упругой постановке [6]. Для уточнения модели и придания нелинейных упругопластических свойств стержневым элементам [7] следует заменить алгоритм формирования локальных матриц жесткости C11, C12, C21, C22, C в классе *TRod*.

С точки зрения применимости матричной формы метод сил аналогичен методу перемещений, однако для сложных стержневых структур программирование алгоритма построения основной системы метода сил вызывает некоторые трудности. В свою очередь, неизвестными метода перемещений являются кинематические параметры, что затрудняет его взаимосвязь с общим уравнением динамики, где в качестве неизвестных также используются параметры кинематики. Наиболее приемлемым решением в данном случае следует считать матричную форму метода перемещений, в которой построение матрицы податливости частично ведется по алгоритму метода сил [3]. Полученные таким образом силовые факторы уже учитывают внешнюю узловую и внеузловую нагрузки, а также кинематическое возмущение в тех ее узлах, для которых интегрированием основного уравнения динамики [8] были установлены текущие координаты и ориентация.

Приведем реализацию основных функций в объявленных классах. Нумерация элементов в массивах принята с нуля. Силы реакций на заданные перемещения узлов стержня входят в правую часть уравнений (1) и вычисляются умножением общей матрицы жесткости системы на эти перемещения. Функция, возвращающая результат этого произведения, имеет вид

```
function TConstruction.NodeReaction :TVector; {силы реакций в узлах}
var i: Integer;
   R, T :TMatrix;
   V :TVector;
begin
   result := Matrix2Vector(MultM(GetSTIFF, Translation));
end;
```

Общая локальная матрица жесткости стержневого элемента размером 12×12 для обоих концов описывается блочной матрицей

$$C=\left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right]. \qquad (2)$$

Функция, возвращающая этот блок матрицы, имеет вид

```
function TRod.C11 :TMatrix;
```

```
var M :TMatrix;
   L2, L3 :TElement;
begin
   L2 := L*L; L3 := L2*L;
   M := TMatrix.CreateTemp(6, 6);
   M[0, 0] := E*A/L;
   M[1, 1] := 12.0*E*Jz/L3; M[1, 5] := 6.0*E*Jz/L2;
   M[2, 2] := 12.0*E*Jy/L3; M[2, 4] := -6.0*E*Jy/L2;
   M[3, 3] := G*Jx/L;
   M[4, 2] := -6.0*E*Jy/L2; M[4, 4] := 4.0*E*Jy/L;
   M[5, 1] := 6.0*E*Jz/L2;  M[5, 5] := 4.0*E*Jz/L;
   result := M;
end;
```

Остальные три блока формируются аналогично. Общую матрицу жесткости возвращает функция C:

```
function TRod.C :TMatrix;
begin
   result := Augment(Stack(C11, C21), Stack(C12, C22));
end;
```

Вспомогательные функции *Augment* и *Stack* представляют собой аналоги одноименных функций из математического пакета *Mathcad* для объединения матриц [5].

Функция, реализующая построение общей матрицы жесткости системы:

```
function TConstruction.GetSTIFF :TMatrix;
var i, j, b, e :Integer;
   S :TMatrix;
   R :TRod;
begin
   S := TMatrix.CreateTemp(NodeCount, NodeCount);
   for i:=0 to NodeCount – 1
   do for j:=0 to NodeCount – 1
      do S.MatrixZeroOrigin[i, j] := Zero(6);
   for j:=0 to RodCount – 1
   do begin
     R := Rods[j];
     b := R.begin_node;
     e := R.end_node;
     S.MatrixZeroOrigin[b, b] := AddM(R.C11, S.MatrixZeroOrigin[b, b]);
     S.MatrixZeroOrigin[e, e] := AddM(R.C22, S.MatrixZeroOrigin[e, e]);
     S.MatrixZeroOrigin[b, e] := AddM(R.C12, S.MatrixZeroOrigin[b, e]);
     S.MatrixZeroOrigin[e, b] := AddM(R.C21, S.MatrixZeroOrigin[e, b]);
   end;
   result := S;
end;
```

Функция *AddM* складывает две исходные матрицы и возвращает результат сложения.

Для получения перемещений узлов используем функцию *Translation*, которая обнуляет перемещения для кинематических закреплений, а остальные перемещения вычисляет с помощью функции *Translation_DOF*

```
function TConstruction.Translation: TVector;
var R, T, V :TVector;
   i, ind :Integer;
begin
R := TVector.CreateTemp(6*n); R.RowNameCreate;
T := Translation_DOF;
V := DOF;
ind := 0;
for i:=0 to 6*n-1
do begin
   R.RowName[i] := V.RowName[i];
   if (V[i] = 1.0)
   then R[i] := 0.0
   else begin
     R[i] := T[ind];  Inc(ind);
   end;
end;
result := R;
T.Free; V.Free;
end;
```

Функция *Translation_DOF* возвращает перемещения узлов, свободных от кинематических закреплений, путем умножения матрицы податливости, из которой удалены строки и столбцы, соответствующие этим закреплениям, на вектор узловых нагрузок

```
function TConstruction.Translation_DOF: TVector;
var i :Integer;
   D :TVector;
begin
   result := Matrix2Vector(MultM(InvM(STIFF_DOF), FT_DOF));
   result.RowNameCreate;
   D := DOF_DOF;
   for i:=0 to D.Rows-1
   do result.RowName[i] := D.RowName[i];
   D.Free;
end;
```

Функция, возвращающая вектор узловых нагрузок по направлениям, свободным от кинематических закреплений:

```
function TConstruction.FT_DOF :TVector;
var F, V :TVector;
   i :Integer;
begin
F := FT;
V := DOF;
   for i:=6*NodeCount-1 downto 0
   do if (V[i] = 1.0)
   then F.DeleteRow(i);
V.Free;
result := F;
end;
```

Функция, возвращающая матрицу кинематических закреплений с булевыми признаками:

```
function TConstruction.DOF :TVector;
var V, D :TVector;
   i, j :Integer;
begin
V := nil;
for i:=0 to NodeCount – 1
do begin
   D := Nodes[i].Disp.AsVector;
   for j:=0 to 5
   do D.RowName[j] := D.RowName[j] + IntToStr(i);
   V := Stack(V, D);
end;
   V.Name := 'DOF';
   result := V;
end;
```

Функция, возвращающая общую матрицу жесткости системы с учетом кинематических закреплений (при наличии закрепления узла столбец и строка в исходной матрице жесткости удаляются):

```
function TConstruction.STIFF_DOF :TMatrix;
var S :TMatrix;
   V :TVector;
   i :Integer;
begin
   S := STIFF;
   V := DOF;
   for i:=6*NodeCount-1 downto 0
   do begin
   if (V[i] = 1.0)
   then begin
     S.DeleteCol(i); S.DeleteRow(i);
   end;
   end;
   V.Free;
   result := S;
end.
```

Выводы

Различные формы общего уравнения динамики позволяют с высокой степенью точности определить силы инерции и кинематические параметры элементов механической системы, а в сочетании с методом конечных элементов эффективно решать задачи динамики строительных конструкций как совместной системы твердых и упругих тел. Отметим, что универсальные алгоритмы метода конечных элементов, положенные в основу статического расчета многих систем автоматизированного проектирования, были реализованы в виде подпрограмм определения усилий в стержнях для уравнений движения дискретных элементов. Приведенный алгоритм, описанный с помощью языка объектно-ориентированного программирования в терминах классов и функций, может быть основой для программной реализации профильного расчетного пакета.

Литература

1. Неймарк Ю.И. Динамика неголономных систем // Ю.И. Неймарк, Н.А. Фуфаев / – М.: Гл. ред. физ.-мат. лит., изд-во «Наука», 1967. – 529 с.
2. Распопов А.С., Русу С.П., Артемов В.Е. Применение уравнений Эйлера-Лагранжа к решению задачи динамики системы «мост-поезд» // Вестник Днепр. нац. ун-та жел. дор. тр-та. – Дн-ск, 2007. – Вып. 16. – С. 109-114.
3. Розин Л.А. Стержневые системы как системы конечных элементов. –Л.: Изд-во Ленингр. ун-та, 1975. 237с.
4. Сегерлинд Л. Применение метода конечных элементов. – М.: Мир, 1979. – 392 с.
5. Бачурин Л.Л. Решение плоской задачи механики деформируемого твердого тела методом конечных элементов в пакете Mathcad // Exponenta Pro / Методы, алгоритмы, программы. – № 3 (3), 2003. – С. 28-33.
6. Белл Ф.Дж. Экспериментальные основы механики деформируемых твердых тел. В 2-х ч. Ч. 1. Малые деформации. – М.: Наука, 1984. – 600 с.
7. Белл Ф.Дж. Экспериментальные основы механики деформируемых твердых тел. В 2-х ч. Ч. Ч. 2. Конечные деформации. – М.: Наука, 1984. – 432 с.
8. Вильке В.Г. Теоретическая механика. – СПб.: Изд-во «Лань», 2003. – 304 с.