МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ

Численные методы расчета строительных конструкций

Код направления подготовки / специальности	08.04.01
Направление подготовки / специальность	Строительство
Наименование ОПОП (профиль/магистерская программа/программа аспирантуры)	Промышленное и гражданское строительство
Год начала подготовки	2022
Уровень образования	магистратура
Форма обучения	очная, заочная

РАЗДЕЛ 1

Оглавление

Лекция 1

КРАЕВАЯ ЗАДАЧА

Введение

В настоящее время применение численных методов и методов компьютерного моделирования не ограничивается только естественнонаучными областями знания, но и активно востребовано в сфере, традиционно относимой к гуманитарной. Это - история, политика, экономика, психология и некоторые другие области знаковой деятельности человека.

Активное продвижение математических методов и технологий во все сферы знаковой деятельности человека стало возможным с появлением компьютера, т.е. с появлением компьютерной или, более точно, информационной индустрии. Традиционное использование в науке компьютера, как универсального вычислительного устройства получило в последнее время новое содержание в рамах таких понятий, как виртуальная реальность и интерактивный интерфейс. Современные вычислительные мощности и средства программирования позволяют в ряде случаев не просто решить ту или иную задачу, но и построить нечто большее - виртуальную реальность, в которой исследователь может осознать решение задачи во всем

возможном многообразии. Привлечение средств интерактивного интерфейса позволяет активно манипулировать доступным многообразием для получения оптимального в том или ином смысле решения исходной задачи. В этом контексте курс лекций построен таким образом, чтобы студенты могли непосредственно использовать такой широко известный пакет программирования, как MATLAB, для активного овладения вычислительными методами.

В курсе строительной информатики будут рассмотрены современные методы численного и компьютерного моделирования для решения прикладных задач строительной отрасли и их реализациями в системе MATLAB.

1.1 Обыкновенные дифференциальные уравнения

Обыкновенные дифференциальные уравнения занимают видное место в науке. Они широко используются в механике при расчете различного рода движений, в других разделах физики, например, при расчетах электрических цепей, в химической кинетике при анализе баланса реагентов, в популяционной биологии (модели типа хищник-жертва), в строительстве при расчете сооружений, в экономике, в политике и во множестве других областях.

В общем и целом обыкновенные дифференциальные уравнения являются одним из самых востребованных инструментов описания и расчета самых разнообразных явлений и процессов в науке, технике и обществоведении.

Обыкновенное дифференциальное уравнение - это уравнение, содержащее аргумент, искомую функцию этого аргумента и ее производные различных порядков:

$$F(x, y, y', y'', .., y^{(n)}) = 0 \qquad (1.1)$$

Порядок дифференциального уравнения — это наивысший порядок производной искомой функции в данном уравнении.

Решением дифференциального уравнения называется функция, которая при подстановке в дифференциальное уравнение обращает последнее в верное равенство.

Частным случаем (1.1) является дифференциальное уравнение первого порядка, разрешенное относительно первой производной:

$$y'=f(x,y) \qquad (1.2)$$

Если неизвестная функция ***y*** является вектором, то тогда задача сводится к системе дифференциальных уравнений.

При этом известно, что дифференциальные уравнения и системы произвольного порядка могут быть сведены к системам дифференциальных уравнений первого порядка, используя введение вспомогательных функций, количество которых соответствует порядку дифференциального уравнения.

Например, для дифференциального уравнения второго порядка

$$y''=f(x,y,y')$$

можем ввести новые функции:

$$y_1 = y, \quad y_2 = y_1' = y',$$

тогда дифференциальное уравнение второго порядка сводится к дифференциальным уравнениям первого порядка

$$\begin{cases} y_1' = y_2 \\ y_2' = f(x, y_1, y_2). \end{cases}$$

Известно, что дифференциальное уравнение n-го порядка имеет множество решений, которые в общем случае зависят от n параметров. Для выделения единственного решения необходимо наложить n дополнительных условий.

Расчетные схемы большинства задач расчета сооружений математически представлены в виде краевых задач для дифференциальных уравнений.

Свое название краевые задачи получили вследствие задания условий на краях отрезка [a,b].

Рассмотрим несколько примеров.

Пример 1. Изгибающий момент в балке.

Рассмотрим балку длиной ℓ с шарнирным опиранием, находящуюся под действием нагрузки с эпюрой $q(x)$:

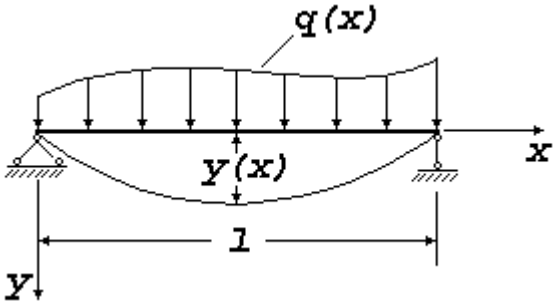

Рис. 1.1

Изгибающий момент $M(x)$ определяется из решения краевой задачи:

$$\begin{cases} M''(x) = -q(x), \quad x \in (0,\ell), \\ \left.\begin{aligned} M(0) = 0 \\ M(\ell) = 0 \end{aligned}\right\} - \textit{краевые условия}. \end{cases} \tag{1.3}$$

Замечание. Для данного случая из курса математики известно аналитическое решение:

$$M(x) = -\int_0^\ell G(x,\xi) q(\xi) d\xi,$$

где $G(x,\xi) = \frac{1}{2}\left(|x-\xi| - (x+\xi)\right) + \frac{x\xi}{\ell}$ – так называемая, функция Грина для задачи (1.3).

Пример 2. Прогиб сжато-изогнутого стержня (балки).

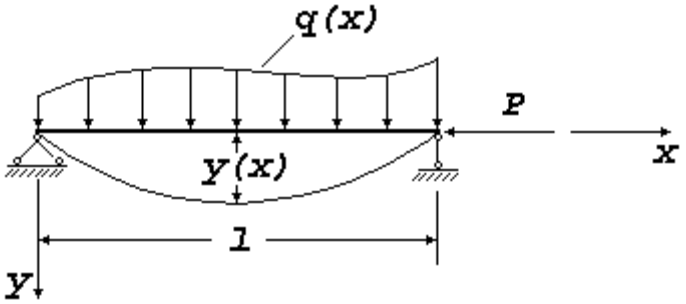

Рис. 1.2

В этом случае (рис. 1.2) в дополнение к нагрузке $q(x)$ приложена сжимающая сила P. Из курса сопротивления материалов известно, что прогиб $y(x)$ удовлетворяет уравнению

$$EJ(x)\cdot y''(x) = M^*(x),$$

где $EJ(x)$ – жесткость балки,

$$M^*(x) = M(x) + Py,$$

$M(x)$ – решение задачи в примере 1.

Из курса высшей математики, а также из физических соображений, известно, что существует бесконечное множество решений приведенного уравнения, если не учитывать поведение функции $y(x)$ на краях. В данном случае из физических условий получаем следующую краевую задачу

$$\begin{cases} y'' - \dfrac{P}{EJ(x)}y = \dfrac{M(x)}{EJ(x)}, \quad x \in (0,\ell), \\ \left.\begin{matrix} y(0) = 0 \\ y(\ell) = 0 \end{matrix}\right\} - \textit{краевые условия.} \end{cases} \tag{1.4}$$

Отметим, что не существует аналитического решения этой задачи для произвольной функции $EJ(x)$, т.е. может идти речь только о численном решении.

1.2. Общий вид краевой задачи на отрезке (a,b) для дифференциального уравнения второго порядка

$$\begin{cases} y'' + P(x)y' + q(x)y = f(x),\ x \in (a,b), \\ \alpha_1 y(a) + \beta_1 y'(a) = \gamma_1 \\ \alpha_2 y(b) + \beta_2 y'(b) = \gamma_2 \end{cases} - \text{краевые условия.} \qquad (1.5)$$

Здесь все величины, кроме $y(x)$, предполагаются заданными.

Отметим еще раз, что существует некоторое бесконечное множество функций, удовлетворяющих приведенному дифференциальному уравнению, и лишь наличие области изменения x ($x \in (a,b)$ – геометрия конструкции) и условий на краях области обеспечивает единственность решения. Однако и это решение (хотя оно и единственное) в общем случае нельзя представить в аналитической форме. Поэтому, как правило, задача решается численным методом.

1.3. Метод конечных разностей

Это наиболее простой и при этом достаточно эффективный способ решения краевой задачи. Суть его состоит в следующем.

Разобьем отрезок (a,b) на $(N-1)$-отрезков (рис.1.3).

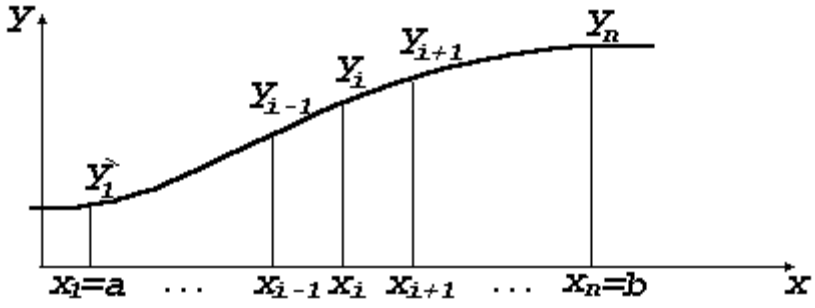

Рис. 1.3.

Введем обозначения:

x_i – координаты точек разбиения,

i – номер точки разбиения ($i = 1,2,...,N$),

$h_i = x_{i+1} - x_i$ – длина i -го отрезка (шаг разбиения),

$\tilde{h}_i = \dfrac{h_i + h_{i-1}}{2} = \dfrac{x_{i+1} - x_{i-1}}{2}$ – «средний» шаг,

$y_i = y(x_i)$, при этом $y_1 = y(a)$, $y_n = y(b)$,

$$P_i = P(x_i), \quad q_i = q(x_i), \quad f_i = f(x_i).$$

Производные в i-ой точке заменим разностными соотношениями:

$$y_i' \approx \frac{y_{i+1} - y_i}{h_i} - \text{правая разность},$$

$$y_i' \approx \frac{y_i - y_{i-1}}{h_{i-1}} - \text{левая разность}, \qquad (1.6)$$

$$y_i' \approx \frac{y_{i+1} - y_{i-1}}{2\tilde{h}_i} - \text{центральная разность}.$$

При $h \to 0$, в соответствии с определением производной, все три величины будут стремиться к $y'(x_i)$. Использование той или другой из них зависит от конкретной ситуации.

Вторая производная в i-ой точке может быть приближенно представлена разностным отношением в виде разности от первых, например,

$$y_i'' \approx \left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right) \Big/ \tilde{h}_i ,$$

при $h_i = h = const$

$$y_i'' \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \qquad (1.7)$$

Здесь предложена наиболее часто употребляемая формула второй разности.

Пользуясь приведенными обозначениями и формулами, можно представить задачу (1.5) в каждой i-ой точке относительно величин y_i следующим образом:

$$\begin{cases} \alpha_1 y_1 + \beta_1 \dfrac{y_2 - y_1}{h_1} = \gamma_1, \quad i = 1, \\ \left(\dfrac{y_{i+1} - y_i}{h_i} - \dfrac{y_i - y_{i-1}}{h_{i-1}} \right) \Big/ \tilde{h}_i + P_i \dfrac{y_{i+1} - y_{i-1}}{2\tilde{h}_i} + q_i y_i = f_i, \quad i = 2,3,\ldots,N-1 \\ \alpha_2 y_N + \beta_2 \dfrac{y_N - y_{N-1}}{h_{N-1}} = \gamma_2, \quad i = N. \end{cases} \qquad (1.8)$$

Это и есть разностный аналог краевой задачи (1.3). Здесь N неизвестных y_i и N уравнений. Приводя подобные члены, получим:

$$\begin{cases} a_{11}y_1 + a_{12}y_2 = b_1 & i = 1, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{i,i-1}\,y_{i-1} + a_{i,i}\,y_i + a_{i,i+1}\,y_{i+1} = b_i, & i = 2,3,\ldots,N-1, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{N,N-1}y_{N-1} + a_{N,N}y_N = b_N & i = N. \end{cases} \qquad (1.9)$$

или в матричной записи

$$A\bar{y} = \bar{b}, \qquad (1.10)$$

где

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & & & & & \vdots \\ 0 & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & a_{i,i-1} & a_{i,i} & a_{i,i+1} & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & & \ddots & \ddots & a_{N-1,N} \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & a_{N,N-1} & a_{N,N} \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_i \\ \vdots \\ \vdots \\ y_N \end{bmatrix}, \quad \bar{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_i \\ \vdots \\ \vdots \\ b_N \end{bmatrix},$$

при этом элементы матрицы и вектора правой части определяются по следующим формулам:

$$a_{11} = \alpha_1 - \frac{\beta_1}{h_1}, \quad a_{12} = \frac{\beta_1}{h_1}, \quad b_1 = \gamma_1,$$

$$a_{i,i-1} = \left(\frac{1}{h_{i-1}} - \frac{p_i}{2}\right)/\tilde{h}_i, \quad a_{ii} = q_i - \frac{2}{h_i h_{i-1}}, \quad a_{i,i+1} = \left(\frac{1}{h_i} + \frac{p_i}{2}\right)/\tilde{h}_i, \quad b_i = f_i,$$

$$i = 2,3,\ldots,N-1,$$

$$a_{N,N-1} = -\frac{\beta_2}{h_{N-1}}, \quad a_{N,N} = \alpha_2 + \frac{\beta_2}{h_{N-1}}, \quad b_N = \gamma_2.$$

Отметим, что A – трехдиагональная матрица.

После решения системы (1.8) получим приближенное решение задачи (1.5) в i-х точках. Соединяя y_i ломаной линией, получаем приближенное решение $y(x)$ во всех точках области (a,b).

Точность решения задачи (1.5) зависит от точности аппроксимации значений производных разностями, которые, в свою очередь, зависят от величины шага разбиения.

Рассмотрим краевую задачу.

$$\begin{cases} y'' - \dfrac{40}{1+4x(1-x)}\, y = -2.4\left(2 - \dfrac{40}{1+4x(1-x)}\, x(1-x)\right); \quad x \in (0,1), \\ \left.\begin{aligned} y(0) &= 0 \\ y(1) &= 0 \end{aligned}\right\} - \textit{краевые} \quad \textit{условия} \end{cases}$$

М-файл для этой задачи:

```
dl=1; y0=0; yl=0;
n=9;
h=dl/(n-1); xi=(0:h:dl)';
x=xi(2:n-1);
p=-40/(dl^2+4*x.*(dl-x));
f=-2.4*(2-p.*x.*(dl-x));
t=ones(n-2,1);
%формируем трехдиагональную матрицу
a=diag([t;0],-1)+diag([1;h^2*p-2;1])+diag([0;t],1);

%формируем вектор правой части
b=[y0;h^2*f;yl];
%решаем систему линейных уравнений
y=a\b;
%вывод результатов
disp('матрица системы разностных уравнений')
for i=1:n
    fprintf('%8.3f',a(i,:)),fprintf('\n')
end
disp('вектор правой части системы разностных уравнений')
fprintf('%8.3f',b),fprintf('\n')
disp('решение y:'),fprintf('%8.3f',y),fprintf('\n')
%вывод графика
plot(xi,y),grid on
```

Решение

```
матрица системы разностных уравнений
   1.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000
   1.000   -2.435    1.000    0.000    0.000    0.000    0.000    0.000    0.000
   0.000    1.000   -2.357    1.000    0.000    0.000    0.000    0.000    0.000
   0.000    0.000    1.000   -2.323    1.000    0.000    0.000    0.000    0.000
   0.000    0.000    0.000    1.000   -2.313    1.000    0.000    0.000    0.000
   0.000    0.000    0.000    0.000    1.000   -2.323    1.000    0.000    0.000
   0.000    0.000    0.000    0.000    0.000    1.000   -2.357    1.000    0.000
   0.000    0.000    0.000    0.000    0.000    0.000    1.000   -2.435    1.000
   0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000    1.000


вектор правой части системы разностных уравнений
   0.000   -0.189   -0.236   -0.256   -0.263   -0.256   -0.236   -0.189    0.000
решение y:
   0.000    0.262    0.450    0.562    0.600    0.562    0.450    0.262    0.000
```

График решения на рисунке 1.4

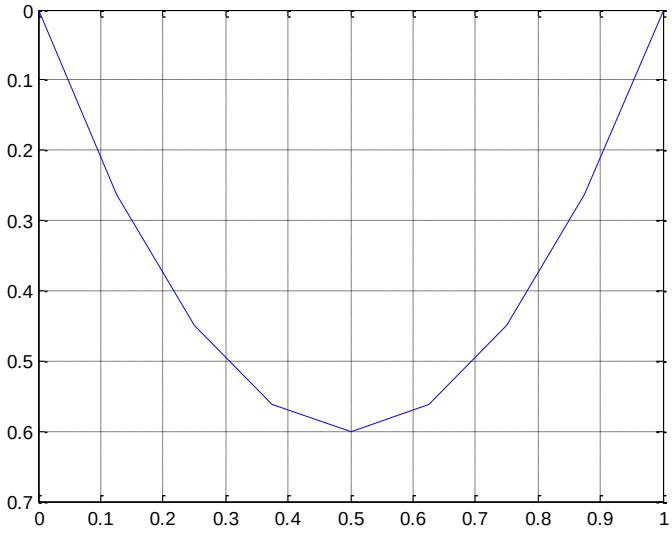

Рис. 1.4 График решения краевой задачи

1.3. Порядок аппроксимации производных

Для упрощения оценок предположим, что $h_i=h=const$. По формуле **Тейлора**:

$$f_{i\pm1} = f(x_i \pm h) = f(x_i) \pm hf'(x_i) + \frac{h^2}{2} f''(x_i) \pm \frac{h^3}{6} f'''(x_i) + \frac{h^4}{24} f^{(4)}(x_i) + O(h^5),$$

где

$$\left| O(h^k) \right| \le Ch^k,\ C = const.$$

Тогда для правой разности получим

$$\frac{f_{i+1} - f_i}{h} = \frac{f(x_i) + hf'(x_i) + \frac{h^2}{2} f''(x_i) + \ldots - f(x_i)}{h} =$$

$$= f'(x_i) + \frac{h}{2} f''(x_i) + \ldots = f'(x_i) + O(h) - \textit{аппроксимация первого порядка}.$$

Аналогично для левой разности

$$\frac{f_i - f_{i-1}}{h} = f'(x_i) + \frac{h}{2} f''(x_i) + \ldots = f'(x_i) + O(h),$$

т.е. также имеет место первый порядок аппроксимации.

Рассмотрим также центральную разность

$$\frac{f_{i+1} - f_{i-1}}{2h} = \frac{f(x_i) + hf'(x_i) + \frac{h^2}{2} f''(x_i) + \frac{h^3}{6} f'''(x_i) + \ldots}{2h} -$$

$$- \frac{f(x_i) - hf'(x_i) + \frac{h^2}{2} f''(x_i) - \frac{h^3}{6} f'''(x_i) + \ldots}{2h} =$$

$$= f'(x_i) + \frac{h^2}{6} f'''(x_i) + \ldots = f'(x_i) + O(h^2) \quad - \textbf{аппроксимация второго порядка}.$$

Следовательно, центральная разность, как правило, является более точной аппроксимацией первой производной.

Установим порядок аппроксимации для второй производной

$$\frac{f_{i+1}-2f_i+f_{i-1}}{h^2}=$$

$$=\frac{f(x_i)+hf'(x_i)+\frac{h^2}{2}f''(x_i)+\frac{h^3}{6}f'''(x_i)+\frac{h^4}{24}f^{(4)}(x_i)+\ldots}{h^2}-$$

$$-2\frac{f(x_i)}{h^2}+$$

$$+\frac{f(x_i)-hf'(x_i)+\frac{h^2}{2}f''(x_i)-\frac{h^3}{6}f'''(x_i)+\frac{h^4}{24}f^{(4)}(x_i)+\ldots}{h^2}=$$

$$=f''(x_i)+\frac{h^2}{12}f^{(4)}(x_i)+\ldots=f''(x_i)+O\left(h^2\right)$$ – **аппроксимация второго порядка**.

Подобным образом можно установить порядок аппроксимации для любой разностной формулы.

1.4. Решение краевой задачи в MATLAB

В системе MATLAB существуют функции для решения дифференциальных уравнений.

Для решения обыкновенных дифференциальных уравнений первого порядка (1.11) в системе MATLAB используется функция **bvp4c.**

$$\bar{y}'=\bar{f}(x,\bar{y}), \qquad (1.11)$$

где

$\bar{y}$ - искомая вектор-функция;

$\bar{y}'$ - покомпонентная производная вектор-функции $\bar{y}$ по переменной x;

$\bar{f}(x,\bar{y})$ - заданная вектор-функция правых частей;

$x\in[a,b]$.

Решение задачи ищется в форме сеточной функции: отрезок [a,b] делится точками $x_1=a<x_2,\ldots<x_n=b$ на части, не обязательно равные, и каждой точке x_i ставится соответствие искомое значение y_i. Исходя из начальных значений x_i и y_i, путем аппроксимации соответствующей производной в каждой точке сетки

на основе рассмотренных ранее соотношений метода конечных разностей строится разрешающая система алгебраических линейных уравнений. Решением этой системы и является все значения y_i. В процессе решения задачи сетка может перестраиваться (например, сгущаться), при этом на этапе решения используется якобиан $\partial \bar{f} / \partial \bar{y}$.

Начальные значения x_i и y_i определяются с помощью функции **bvpinit**:

```
solinit=bvpinit(xinit,yinit)
```

где

`xinit` – вектор-строка `xinit(1)=a< xinit(2)<… xinit(n)=b;`

`yinit` – предполагаемые значения для y(i), которые могут задаваться в различных формах:

- в форме вектора, каждая компонента которого `yinit(i)` копируется в качестве предполагаемого решения для всех точек сетки, т.е. `y(i,:)= yinit(i);`
- в форме функции, задаваемой `y=gess(x)`, где `x` - произвольная точка отрезка [a,b], `y-` вектор, длина которого равна порядку системе дифференциальных уравнений, при этом для каждой точки сетки `x(i)` вычисляется вектор предполагаемого решения `y(i,:)=gess(x(i));`

`solinit` – структура, в которой при описанном способе обращения к функции заполняются два поля `solinit.x=xinit` и `solinit.y=y(i,:)`.

простейшая форма обращения к функции **bvp4c**

```
sol= bvp4c(odefun,bcfun,solinit)
```

где

`odefun` – функция, вычисляющая вектор правых частей;

`bcfun` – функция, вычисляющая вектор граничных условий, относительно аргументов `ya` и `yb`, которые являются векторами решений `y` в точках `a` и `b;`

`solinit` — выходная структура функции `bvpinit;`

`sol` — аналогичная структура, содержащая решение краевой задачи, имеющая поля `sol.x, sol.y` и `sol.yp.` Последнее поле содержит значение производной `(sol.y)'` в точках `sol.x.`

Решение краевой задачи чаще всего отображают в виде графика. На данном этапе график будет получен из набора ломанных из-за недостаточности количества точек в `sol.x.` Для решения этой проблемы в системе MATLAB можно воспользоваться функцией `deval`. Эта функция, анализируя информацию, которая содержится в структуре `sol,` строит интерполяционный сплайн Эрмита для заданных точек x, чтобы получить необходимое количество точек для обеспечения необходимой гладкости результата.

Обращение к функции `deval` имеет вид:

```
y= deval(sol,x)
```

где

`sol`— выходная структура функции `bvp4c;`

`x` — вектор пробных точек;

`y`— значение сплайн функции в пробных точках.

Алгоритм решения обыкновенного дифференциального уравнения с последующим выводом результата в виде графика состоит в последовательном использовании трех функций:

1. `bvpinit`
2. `bvp4c`
3. `deval`

Рассмотрим решение краевой задачи в системе MATLAB на примере решения уравнения

$$y^{(VI)} + y = 0, \quad y(0) = y'(0) = y''(0) = -1, \quad y'''(\pi) = y^{(IV)}(\pi) = y^{(V)}(\pi) = 1.$$

Решение задачи предполагает переход от уравнения шестого порядка к представлению его в виде соответствующей системы уравнений шестого порядка.

Ниже приведен соответствующий текст М-файла:

```
% Программа  решения  краевой задачи для уравнения
y"""+y=0
function bvp
a=0;b=pi;
%определяем структуру начальных данных
solinit=bvpinit(linspace(a,b,15),[1 1 1 1 1 1]);
%решаем краевую задачу
sol= bvp4c(@f,@bound,solinit);
x=linspace(0,pi,30);
y=deval(sol,x);
%рисуем решение и 5 первых его производных

plot(x,y(1,:),'-o',x,y(2,:),'-p',x,y(3,:), ...
    '-+',x,y(4,:),'-*',x,y(5,:),'-v',x,y(6,:),'-h');

%вычисление правой части системы уравнений

function dydx=f(x,y)
dydx=[y(2) y(3) y(4) y(5) y(6) -y(1)];
end

%определение краевых условий
 function bnd=bound(ya,yb)
bnd=[ya(1)+1; ya(2)+1; ya(3)+1; yb(4)-1; yb(5)-1; yb(6)-
1];
 end

end
```

График задачи приведен на рисунке 1.5

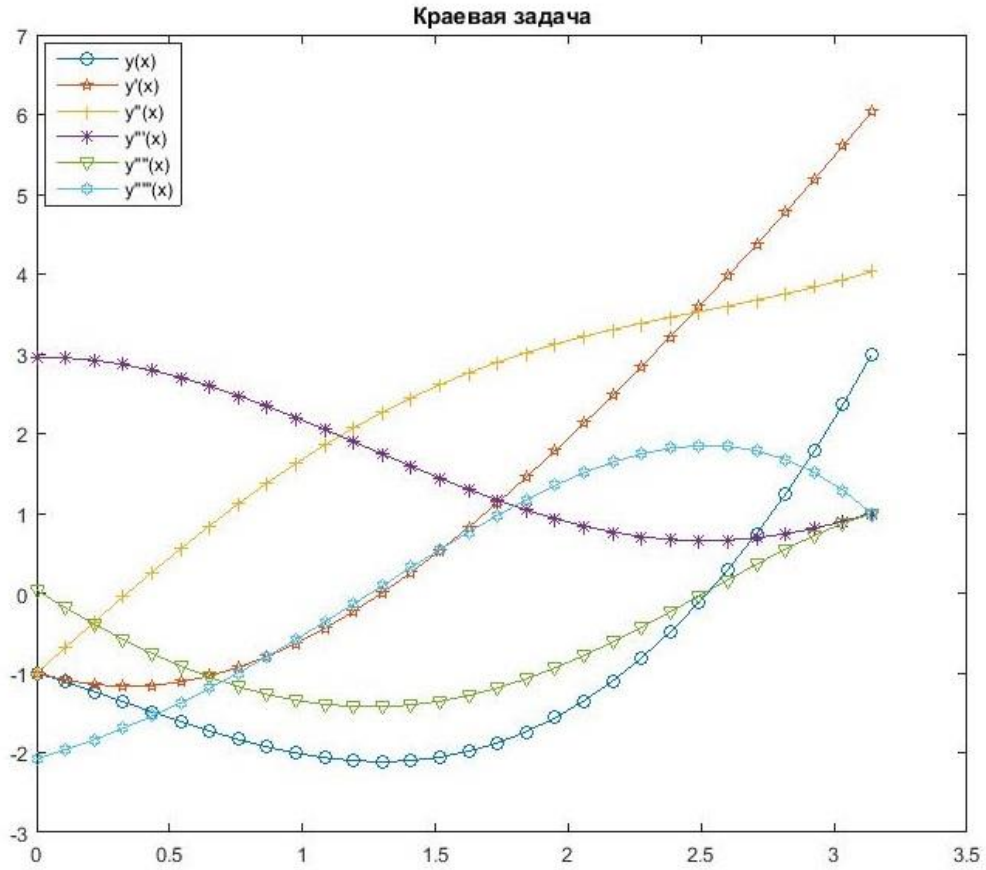

Рис. 1.5 Краевая задача для дифференциального уравнения шестого порядка

Полная форма обращения к функциям `bvpinit` и `bvp4c` представляет следующие возможности:

- учет дополнительных условий, используемых при решении;
- использование дополнительных аргументов в функциях `odefun` и `bcfun`;
- определение значений неизвестных параметров, например, собственных чисел.

Полный синтаксис обращения к функции `bvp4c` имеет вид:

```
sol= bvp4c(odefun,bcfun,solinit,options,P1,P2,…)
```

где `options` — аргумент, позволяющий задать управляющие параметры; `P1,P2,` — дополнительные аргументы для вычисления `odefun` и `bcfun`.

Опишем основные управляющие параметры, которые может задать аргумент `options`:

- параметр `RelTol` задает допустимую относительную погрешность вычислений (по-умолчанию она равна 1e-3);
- параметр `AbsTol` задает допустимую абсолютную погрешность вычислений (по-умолчанию она равна 1e-6). Может быть, и задана вектор-строка абсолютных погрешностей для каждой компоненты решения;
- параметр `FJacobian` задает в аналитическом виде якобиан $\partial\bar{f}/\partial\bar{y}$, где $\bar{f}$ — функция, вычисляемая в `odefun` (значением этого параметра является указанием на М-файл, содержащий функцию, которая вычисляет соответствующую матрицу как функцию переменных `x` и `y`);
- параметр `BCJacobian` задает в аналитическом виде два якобиана $[\partial(bc)/\partial(ya)]$ и $[\partial(bc)/\partial(yb)]$, где `bc(ya,yb)` — функция, вычисляемая в `bcfun` (значением этого параметра является указанием на М-файл, содержащий функцию, которая вычисляет соответствующие матрицы как функции переменных `ya` и `yb)`.

Неизвестный параметр (вектор неизвестных параметров), который вводится с помощью функции `bvpinit`.

Полный синтаксис этой функции:

```
solinit=bvpinit(xinit,yinit, parameters)
```

где

- `parameters`-предполагаемое значение неизвестного параметра (вектор предполагаемых значений неизвестных параметров);
- `solinit` — структура, в которой помимо `solinit.x=xinit` и `solinit.y=y(i,:)` заполняются еще и поле `solinit.parameters=parameters`.

При включении в задачу дополнительных неизвестных параметров, требуется увеличение и количества граничных условий - по одному на каждый

дополнительный параметр. Данные параметры должны быть аргументами функций `odefun` и `bcfun`. Это обязательное правило при включении дополнительных неизвестных параметров независимо от того, используются они реально для вычислений указанных функций или нет. Аналитические якобианы при этом, необходимые в случае задания условий `FJacobian` и/или `BCJacobian`, соответственно также расширяются — $[\partial(f)/\partial(y),\ \partial(f)/\partial(p)]$ и $[\partial(bc)/\partial(ya),\ \partial(bc)/\partial(yb),\ \partial(bc)/\partial(p)]$, где для краткости через `p` обозначен неизвестный параметр.

В выходной структуре `sol` при наличии неизвестных параметров также появляется дополнительное поле `sol.parameters`, содержащее найденные значения неизвестных параметров.

В заключении отметим, что в системе MATLAB имеются также средства для символьного решения дифференциальных уравнений. В частности функция

```
dsolve(expr1 [,expr2,…,cond1,cond2,…,var])
```

позволяет получить символьное решение обыкновенного дифференциального уравнения `expr1` или системы обыкновенных дифференциальных уравнений `expr1, expr2,…,` с граничными условиями `cond1,cond2,….` По умолчанию независимой переменной считается t, но ее можно заменить, присвоив соответствующее значение параметру `var.` Следует иметь в виду, что имя независимой переменной не должно начинаться с `D`, т.к. этот символ обозначает производную по независимой переменной $D = \partial/\partial t$, а $D2 = \partial^2/\partial t^2$

Лекция 2

УСТОЙЧИВОСТЬ СЖАТОГО СТЕРЖНЯ

Введение

Наиболее сложным и достаточно частым видом разрушений конструкции является, так называемая, потеря устойчивости. В курсе сопротивления материалов, как правило, дается математическая формулировка устойчивости стержня, которая представляется следующей расчетной схемой (рис 2.1):

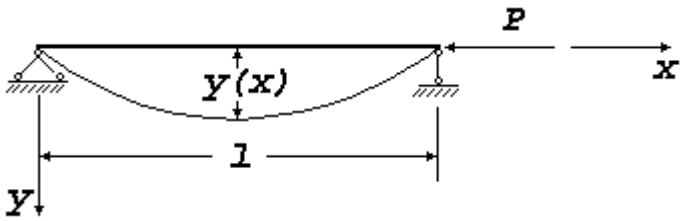

Рис. 2.1.

При действии достаточно небольшой сжимающей силы P на статически определимую балку, ее прогиб равен нулю. Возникает вопрос, существует ли сила P, вызывающая изгиб стержня. Если существует, то она удовлетворяет условиям:

$$\begin{cases} -Ry'' = Py, \quad x \in (0, \ell), \\ y(0) = 0 \\ y(\ell) = 0 \end{cases} - \textit{краевые условия}, \tag{2.1}$$

где

$R = EJ(x)$ – жесткость балки,

P и $y(x) \neq 0$ – являются искомыми величинами.

Очевидно, что при любом P, прогиб $y(x) = 0$ является решением (тривиальное решение). В курсе строительной механики величина P, при которой $y(x) \neq 0$, называется **критической силой**, а $y(x)$ – **формой потери устойчивости**. В математической терминологии величина P называется **собственным значением** оператора краевой задачи (2.1), а $y(x)$ – **собственной функцией** этого оператора.

Аналитическое решение данной задачи существует только при $R = const$, в остальных случая задача решается только численно, в частности, методом конечным разностей.

2.1. Метод конечных разностей

Разобьем отрезок $(0, \ell)$ на $(n+1)$-часть (рис.2.2). Пронумеруем точки: $i = 0,1, \ldots, n, n+1$. Введем следующие обозначения:

x_i – координаты точек разбиения,

i – номер точки разбиения ($i = 0,1,\ldots,n,n+1$),

$h = \dfrac{\ell}{n+1}$ – длина i-го отрезка (шаг разбиения),

$y_i = y(x_i)$, при этом $y_0 = y(0)$, $y_{n+1} = y(\ell)$,

$R_i = R(x_i)$.

Замечание. Предложенная нумерация точек является наиболее удобной для построения разностных формул и алгоритмов численного решения рассматриваемой задачи.

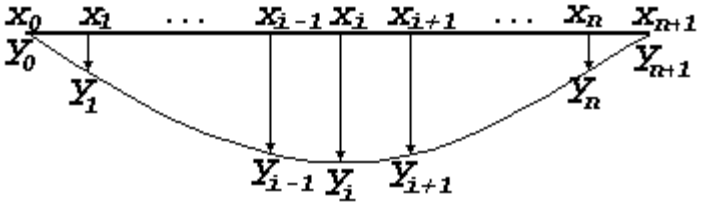

Рис.2.2

Рассматривая задачу (2.1) во внутренних точках разбиения и заменяя вторую производную второй разностью по формуле:

$$y_i'' = y''(x_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$$

получим:

$$-R_i \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = Py_i, \quad i = 1, 2, \ldots, n \qquad (2.2)$$

Для граничных точек:

$$\begin{cases} y_0 = 0, \\ y_{n+1} = 0. \end{cases} \qquad (2.3)$$

Умножая каждое уравнение для внутренних точек на $\frac{h^2}{R_i}$ и учитывая краевые условия (2.3), получим однородную линейную систему уравнений с n неизвестными

$$\begin{cases} 2y_1 - y_2 = P\frac{h^2}{R_1} y_1, & i = 1, \\ -y_{i-1} + 2y_i - y_{i+1} = P\frac{h^2}{R_i} y_i, & i = 2, 3, \ldots, n-1, \\ -y_{n-1} + 2y_n = P\frac{h^2}{R_n} y_n, & i = n. \end{cases} \qquad (2.4)$$

Из курса линейной алгебры известно, что ненулевое решение такой системы существует лишь при некоторых значениях $P = P_{кр}$.

В матричном представлении такая система имеет вид

$$A\bar{y} = PB\bar{y}, \qquad (2.5)$$

где

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & & & \vdots \\ \vdots & -1 & 2 & -1 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & -1 & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} h^2/R_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & h^2/R_2 & & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & h^2/R_{n-1} & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & h^2/R_n \end{bmatrix},$$

$$\bar{y} = [y_1,\ y_2,\ \ldots,\ y_{n-1},\ y_n]^{\mathrm{T}}$$

Таким образом, задача сводится к определению собственных чисел и векторов матрицы $B^{-1}A$. Т.е. каждое собственное число P задачи (2.5) является критической силой, а соответствующий собственный вектор $\bar{y}$ – формой потери устойчивости. На практике основной интерес вызывает минимальная критическая сила $P_{\min}$ и соответствующая ей форма потери устойчивости. Отметим также, что форма потери устойчивости на самом деле является формой разрушения конструкции, поскольку величина $y(x)$ определена с точностью до множителя и, следовательно, может неограниченно возрастать.

2.2. Вычисление минимальной критической силы степенным методом

Умножая матричное уравнение (2.5) на $\frac{1}{P}A^{-1}$, получим

$$\tilde{A}\bar{y} = \lambda\bar{y},$$

где $\tilde{A} = A^{-1}B$ и $\lambda = 1/P$. Следовательно

$$P_{\min} = 1/\lambda_{\max}.$$

Т.е. для определения минимальной критической силы достаточно вычислить максимальное собственное число матрицы $\tilde{A}$. Тогда алгоритм вычислений выглядит следующим образом:

1. - задается $\bar{v}^{(0)}$ – произвольный ненулевой вектор,
2. - вычисляются:

$$\lambda^{(k)} = \left\|\bar{v}^{(k)}\right\|, \quad \bar{y}^{(k)} = \frac{1}{\lambda^{(k)}}\bar{v}^{(k)}, \quad \bar{v}^{(k+1)} = A^{-1}B\bar{y}^{(k)}, \quad k = 1,2,\ldots,$$

3. до тех пор, пока не выполнится условие $\left|\lambda^{(k)} - \lambda^{(k+1)}\right| < \varepsilon$,
4. тогда $P_{\min} \approx \frac{1}{\lambda^{(k)}}$, $\bar{y} \approx \bar{y}^{(k)}$.

2.3 Нахождение собственных значений и собственных векторов в системе MATLAB

Функция [V,D]=**eig**(A) в системе MATLAB возвращает матрицу собственных векторов V (собственные векторы расположены по столбцам) и диагональную матрицу D собственных значений (матрица Жордана (элементы, расположенные на ее главной диагонали есть собственные значения матрицы A); каноническая форма матрицы A), т.е., по сути, определяется разложение Жордана (но при условии отсутствия в матрице Жордана жордановых клеток порядка, большего единицы) и справедливо равенство A*V=V*D. В частном случае, при обращении типа **d=eig(A)** возвращается вектор d, координаты (элементы) которого есть собственные значения (собственные числа) матрицы A.

Создадим симметричную положительно определенную матрицу через функцию **gallery('lehmer',n).** Функция позволяет создать матрицы специального вида, определяемым первым символьным параметром, второй

входной параметр задает размерность нашей матрицы для данного случая. О возможностях функции можно прочитать в документации help gallery.

```
>>A = gallery('lehmer',4)
A =

    1.0000    0.5000    0.3333    0.2500
    0.5000    1.0000    0.6667    0.5000
    0.3333    0.6667    1.0000    0.7500
    0.2500    0.5000    0.7500    1.0000
```

Вычислим собственные значения матрицы A. Результат будет в виде вектор-столбца.

```
>>e = eig(A)
e =

    0.2078
    0.4078
    0.8482
    2.5362
```

Результат можно получить и в виде диагональной матрицы, указав параметр 'matrix'.

```
>>D = eig(A,'matrix')
D =

    0.2078         0         0         0
         0    0.4078         0         0
         0         0    0.8482         0
         0         0         0    2.5362
```

Получить собственные значения и собственные вектора можно, задав два выходных параметра.

```
>> [x,D]=eig(A)

x =

    0.0693   -0.4422   -0.8105    0.3778
   -0.3618    0.7420   -0.1877    0.5322
    0.7694    0.0486    0.3010    0.5614
```

```
   -0.5219   -0.5014    0.4662    0.5088

D =

    0.2078         0         0         0
         0    0.4078         0         0
         0         0    0.8482         0
         0         0         0    2.5362
```

2.4 Алгоритм определения собственных значений оператора краевой задачи на MATLAB

Рассмотрим алгоритм решения задачи для определения минимального собственного значения оператора краевой задачи

$$\begin{cases} -Ry'' = Py, \quad x \in (0,1), \\ \left.\begin{matrix} y(0) = 0 \\ y(1) = 0 \end{matrix}\right\} - \text{краевые условия}, \end{cases}$$

где R(x)=7.5x(1-x).

Система конечно-разностных уравнений этой краевой задачи в матричном виде:

$$A\bar{y} = PB\bar{y}\,.$$

Алгоритм решения задачи определяется последовательностью действий:

1. Выделить память под матрицы с одновременным их обнулением через функцию zeros.
2. Сформировать трехдиагональную матрицу A и диагональную матрицу B.
3. Получить собственные значения и собственные вектора для $B^{-1}A$ через функцию **eig.**
4. Вывести полученные результаты.

Решение задачи состоит в составлении системы при разбиении отрезка $[0, 1]$ на $(N+1)$-частей с постоянным шагом $h = 1/(N+1)$

$$\begin{cases} -7.5x(1-x)y'' = Py, \quad x \in (0,1), \\ \left.\begin{matrix} y(0)=0 \\ y(1)=0 \end{matrix}\right\} - \text{краевые условия.} \end{cases}$$

Текст М-файла, реализующий данный алгоритм:

```
n=input('Введите число точек разбиения n= ');
dl=1;
h=dl/(n+1);
A=zeros(n,n);
B=zeros(n,n);
Y=zeros(n+2,n);
for i=1:n-1
    A(i,i+1)=-1;
    A(i+1,i)=-1;
end
for i=1:n
        x=i*h;
        A(i,i)=2.;
        B(i,i)=h^2/(7.5*(dl-x)*x);
end
disp('Матрица A');
for i=1:n
        fprintf('%6.2f',A(i,:));
        fprintf('\n');
end
disp('Матрица B');
for i=1:n
        fprintf('%6.2f',B(i,:));
        fprintf('\n');
end

[T,J]=eig(inv(B)*A);
P=diag(J);
Y(2:n+1,1:n)=T(1:n,1:n);
disp('Матрица форм потери устойчивости Y');
for i=1:n+2
        fprintf('%6.2f',Y(i,:));
        fprintf('\n');
end
disp(' Вектор критических сил P');
fprintf('%12.4f \n',P);
```

```
Введите число точек разбиения n= 7

Матрица A
  2.00 -1.00  0.00  0.00  0.00  0.00  0.00
 -1.00  2.00 -1.00  0.00  0.00  0.00  0.00
  0.00 -1.00  2.00 -1.00  0.00  0.00  0.00
  0.00  0.00 -1.00  2.00 -1.00  0.00  0.00
  0.00  0.00  0.00 -1.00  2.00 -1.00  0.00
  0.00  0.00  0.00  0.00 -1.00  2.00 -1.00
  0.00  0.00  0.00  0.00  0.00 -1.00  2.00
Матрица B
  0.02  0.00  0.00  0.00  0.00  0.00  0.00
  0.00  0.01  0.00  0.00  0.00  0.00  0.00
  0.00  0.00  0.01  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.01  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.01  0.00  0.00
  0.00  0.00  0.00  0.00  0.00  0.01  0.00
  0.00  0.00  0.00  0.00  0.00  0.00  0.02
Матрица форм потери устойчивости Y
  0.00  0.00  0.00  0.00  0.00  0.00  0.00
  0.03  0.11 -0.24 -0.38 -0.47 -0.21 -0.42
 -0.20 -0.44  0.54  0.33 -0.13 -0.36 -0.48
  0.49  0.55 -0.03  0.49  0.34 -0.45 -0.30
 -0.66 -0.00 -0.54 -0.00  0.54 -0.48  0.00
  0.49 -0.55 -0.03 -0.49  0.34 -0.45  0.30
 -0.20  0.44  0.54 -0.33 -0.13 -0.36  0.48
  0.03 -0.11 -0.24  0.38 -0.47 -0.21  0.42
  0.00  0.00  0.00  0.00  0.00  0.00  0.00
 Вектор критических сил P
    420.0000
    315.0000
    225.0000
    150.0000
     90.0000
     15.0000
     45.0000
```

Для отображения графически форм потери устойчивости добавим в М-файл код

```
x=0:h:1;
for i=1:5
        subplot(7,1,i);
        plot(x,Y(:,i));
end
```

```
subplot(7,1,6);
plot(x,Y(:,7))
subplot(7,1,7);
plot(x,Y(:,6))
```

Результат представлен на рисунке 2.3. Из рисунка видно, что для некоторых форм для отображения в графическом виде недостаточно того количества точек, что использовались при решении задачи.

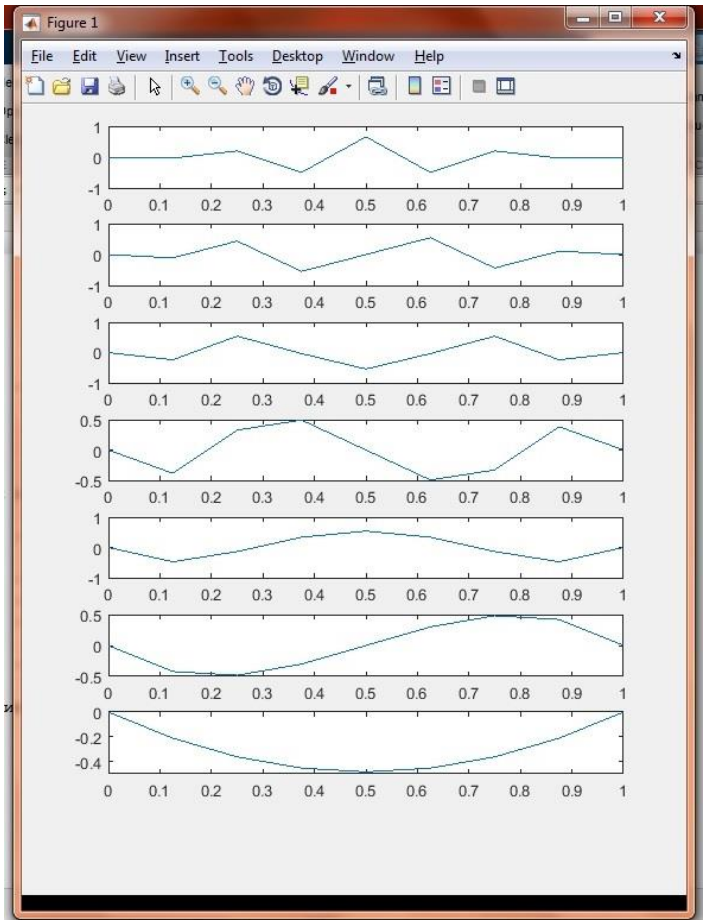

Рис. 2.3 Формы потери устойчивости

Для гладкости отображения можно воспользоваться интерполяцией кубическими сплайнами. **Сплайн** - это непрерывная гладкая функция, которая на отрезках области определения равна полиномам определенной степени, т.е. при таком способе интерполяции точки попарно соединяются отрезками полиномов. На практике используют полиномы невысокой степени, чаще — третьей, т.е. имеет место интерполяция кубическими сплайнами. Таким

образом, аппроксимация происходит не глобально на всем интервале, а по отдельности на каждом частичном интервале между соседними узлами. График сплайна проходит через узлы, в которых обеспечивается стыковка - совпадение значений, совпадение производных для отсутствия изломов, иногда обеспечивается и совпадение вторых производных и т.д. Могут задаваться и краевые условия на концах интервала.

Для выполнения интерполяции кубическими сплайнами можно использовать функцию `spline`, синтаксис которой имеет вид:

```
yy=spline(x,y,xx)
```

В результате будет выполнена интерполяция значений вектора `y`, заданного для значений аргумента, представленного вектором `x`. Функция возвращает вектор `yy`, содержащий значения интерполирующей функции для значений аргумента, заданного вектором `xx`.

Для нашей задачи интерполяция кубическими сплайнами для значений, заданными вектором `x` и столбцом матрицы `Y(:,i)` может быть представлена следующим фрагментом М-файла:

```
x=0:h:1;
h=h/10;
xx=0:h:1;
for i=1:5
        subplot(7,1,i);
        yy=spline(x,Y(:,i),xx);
        plot(xx,yy);
end
    subplot(7,1,6);
    yy=spline(x,Y(:,7),xx);
    plot(xx,yy)
    subplot(7,1,7);
    yy=spline(x,Y(:,6),xx);
    plot(xx,yy)
```

Полученный результат представлен на рисунке 2.4.

Формы потери устойчивости выведены в соответствии со значениями критических сил: от самой максимальной до самой минимальной сверху вниз.

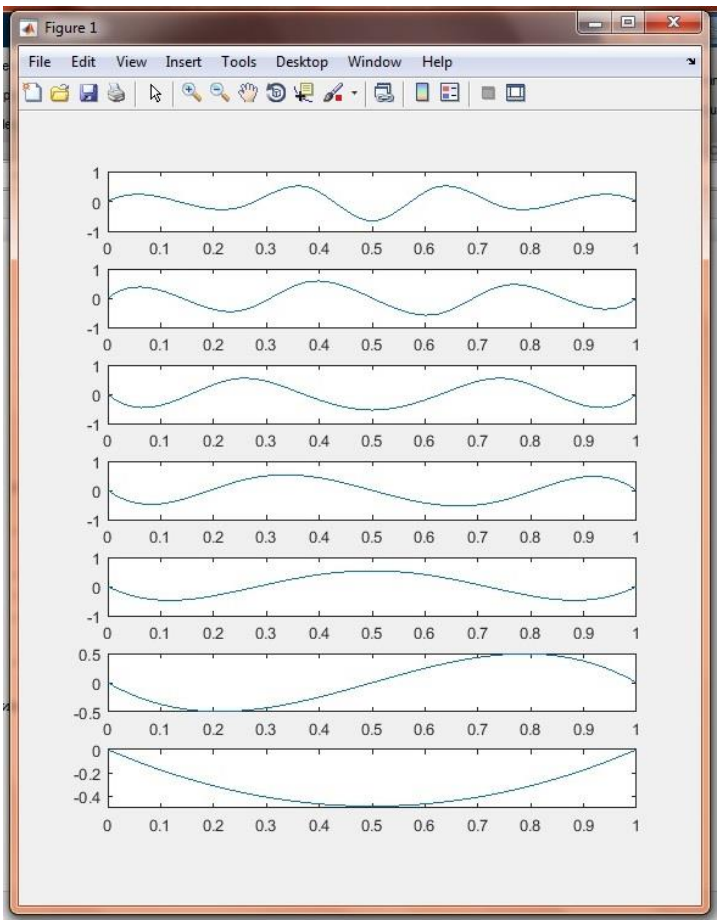

Рис. 2.4 Формы потери устойчивости.

Лекция 3

КРАЕВАЯ ЗАДАЧА ДЛЯ УРАВНЕНИЯ ПУАССОНА

Введение

3.1. Общий вид формулировки краевой задачи для уравнения Пуассона

3.2. Решение задачи Дирихле методом конечных разностей

3.3. Решение задачи Дирихле методом конечных разностей в MATLAB

Введение

Краевая задача для уравнения **Пуассона** является математическим описанием разнообразных технических задач: задачи изгиба мембраны, задачи стационарного распределения температуры в конструкциях, задачи кручения стержня, входит главной составной частью в задачи теории упругости и т.д.

Пример. Кручение стержня.

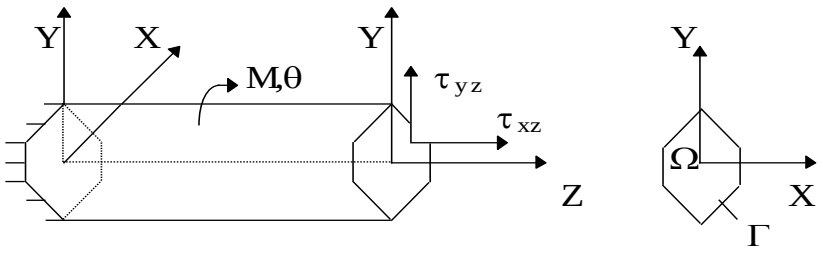

Рис. 3.1

Задача кручения стержня состоит в определении напряженно-деформированного состояния при действии на стержень крутящего момента. При этом часть напряжений известна ($\sigma_x = 0$, $\sigma_y = 0$, $\tau_{xy} = 0$). Неизвестными остаются

$$\tau_x = \tau_{xz} = ?, \quad \tau_y = \tau_{yz} = ?$$

В теории упругости показывается, что из уравнения равновесия и совместности деформаций вытекает, что касательные напряжения выражаются следующими формулами:

$$\tau_x = G\theta\frac{\partial U}{\partial y}, \quad \tau_y = G\theta\frac{\partial U}{\partial x},$$

где

G – модуль упругости сдвига,

θ – угол закручивания (задан),

$U(x, y)$ – функция Прандтля (функция кручения), являющаяся решением краевой задачи:

$$\begin{cases} \nabla^2 U = -2, & (x, y) \in \Omega \\ U_\Gamma = 0 \end{cases},$$

здесь $\nabla^2 = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2}$ – оператор Лапласа.

Замечание. Часто оператор **Лапласа** обозначается в виде Δ.

3.1. Общий вид формулировки краевой задачи для уравнения Пуассона

Пусть краевая задача рассматривается в относительно некоторой области Ω (рис. 3.2).

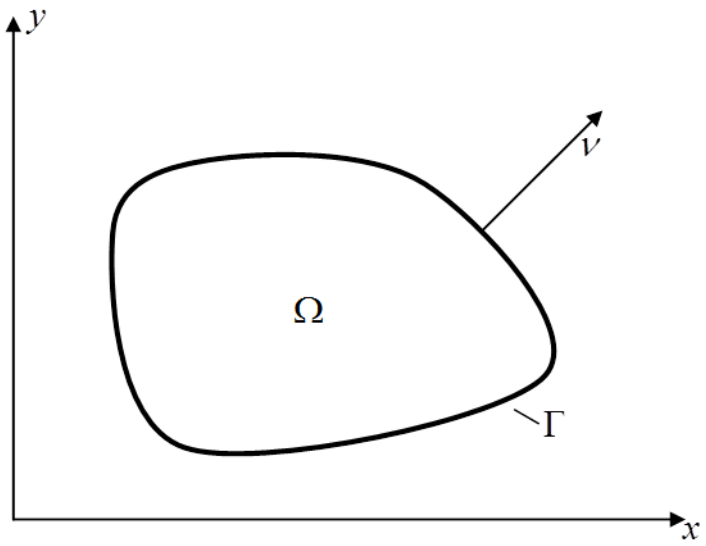

Рис. 3.2

где

$\nu = (\nu_1, \nu_2)$ – вектор нормали к границе области Ω:

$$\nu_1 = \cos(\nu, x), \quad \nu_2 = \cos(\nu, y),$$

$\Gamma = \partial\Omega$ – граница области Ω.

В зависимости от условий на краях области (краевых условий) краевая задача может иметь разные названия. В частности:

$$\begin{cases} \nabla^2 U = F, & (x, y) \in \Omega \\ U_\Gamma = g \end{cases}$$ – **задача Дирихле** (первая краевая задача),

$$\begin{cases} \nabla^2 U = F, & (x, y) \in \Omega \\ \left.\dfrac{\partial U}{\partial \nu}\right|_\Gamma = f \end{cases}$$ – **задача Неймана** (вторая краевая задача),

где

$\dfrac{\partial U}{\partial \nu} = \nu_1 \dfrac{\partial U}{\partial x} + \nu_2 \dfrac{\partial U}{\partial y}$ – производная по нормали.

Если на одной части границы заданы условия задачи Дирихле, а на другой – условия задачи Неймана, тогда таким образом сформулированная задача называется **смешанной** краевой задачей.

3.2. Решение задачи Дирихле методом конечных разностей

Рассмотрим задачу Дирихле в прямоугольной области Ω (рис. 3.3).

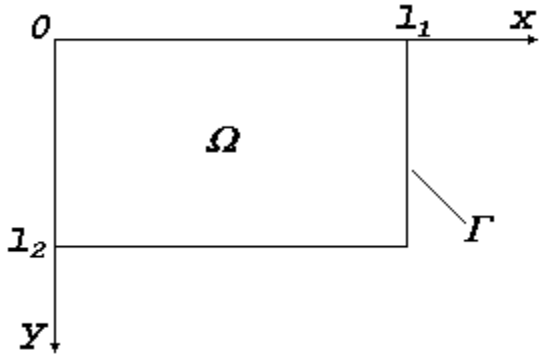

Рис. 3.3

Тогда ее математическую формулировку можно представить в виде:

$$\begin{cases} \nabla^2 U = F, \qquad \Omega = (0 < x < \ell_1,\ 0 < y < \ell_2), \\ U_\Gamma = \varphi(x,y) = \begin{cases} \varphi_1(y), & x = 0 \\ \varphi_2(y), & x = \ell_1 \\ \varphi_3(x), & y = 0 \\ \varphi_4(x), & y = \ell_2, \end{cases} \quad - \textit{краевые условия.} \end{cases} \tag{3.1}$$

Здесь все величины заданы, кроме функции U(x,y) – искомая функция.

Примечание. При решении задачи на ЭВМ естественным является направление оси y вниз, поскольку это соответствует естественному порядку

вывода (печати) на экран и принтер (вывод информации последовательно сверху вниз).

Разобьем исходную прямоугольную область на мелкие прямоугольники с помощью сетки с шагом h_1 по 1-му направлению (по координате x) и с шагом h_2 – по 2-му направлению (по координате y):

$$h_1 = \frac{\ell_1}{N_1} \quad \text{и} \quad h_2 = \frac{\ell_2}{N_2}$$

где N_1+1 и N_2+1 – число узлов сетки по 1-му и 2-му направлению, соответственно (рис. 3.4).

Будем рассматривать задачу (3.1) в узлах сетки. Каждый узел сетки имеет номер, определяемый двумя величинами (i, j):

$j = 0,1,\ldots,N_1$ – номер узла по направлению оси x,

$i = 0,1,\ldots,N_2$ – номер узла по направлению оси y

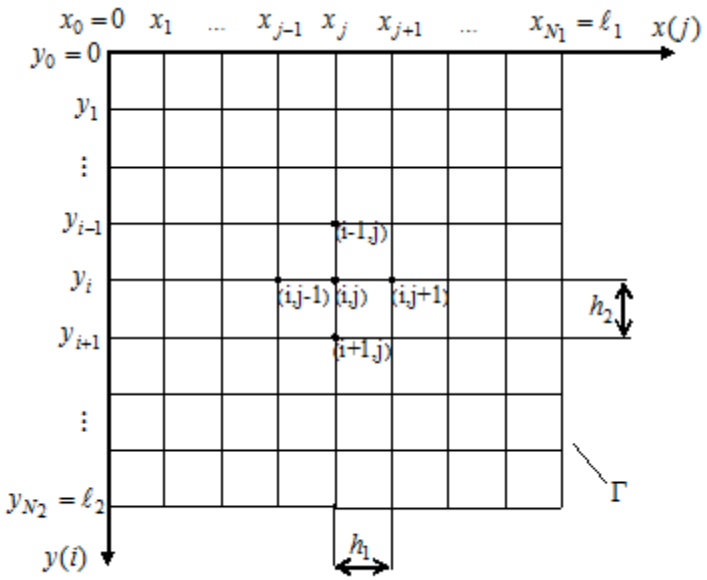

Рис. 3.4

Обозначим

$$U_{ij} = U(x_j, y_i), \quad F_{ij} = F(x_j, y_i), \quad \varphi_{ij} = \varphi(x_j, y_i).$$

Вторые производные для внутренних узлов сетки (i, j) по каждому направлению заменим вторыми разностями, т.е.

$$\frac{\partial^2 U(x_j, y_i)}{\partial x^2} \approx \frac{U_{i,j-1} - 2U_{ij} + U_{i,j+1}}{h_1^2},$$

$$\frac{\partial^2 U(x_j, y_i)}{\partial y^2} \approx \frac{U_{i-1,j} - 2U_{ij} + U_{i+1,j}}{h_2^2}.$$

Тогда краевая задача в конечных разностях примет вид

Для внутренних узлов

$$\frac{U_{i,j-1} - 2U_{ij} + U_{i,j+1}}{h_1^2} + \frac{U_{i-1,j} - 2U_{ij} + U_{i+1,j}}{h_2^2} = F_{ij} \qquad (3.2)$$

$j = 1,2,\ldots,N_1 - 1, \quad i = 1,2,\ldots N_2 - 1,$

для граничных узлов: $U_{ij} = \varphi_{ij} = \begin{cases} \varphi_1(y_i), & j = 0, \\ \varphi_2(y_i), & j = N_1, \\ \varphi_3(x_j), & i = 0, \\ \varphi_4(x_j), & i = N_2. \end{cases}$

Т.е. для определения U_{ij} во внутренних точках достаточно решить систему из $(N_1 - 1) \times (N_2 - 1)$ уравнений с $(N_1 - 1) \times (N_2 - 1)$ неизвестными, используя заданные значения U_{ij} на границе.

С алгоритмической точки зрения наиболее простыми способами решения такой системы являются итерационные, в частности, метод **Зейделя** и метод простой итерации. Для этого, выделив из уравнений (3.2) диагональный член U_{ij}, получим

$$U_{ij} = \left(\frac{U_{i,j-1} + U_{i,j+1}}{h_1^2} + \frac{U_{i-1,j} + U_{i+1,j}}{h_2^2} - F_{ij} \right) \Big/ \left(\frac{2}{h_1^2} + \frac{2}{h_2^2} \right).$$

Далее, задав начальное приближение $U_{ij}^{(0)} = 0$ для внутренних точек и присвоив заданные значения $U_{ij} = \varphi_{ij}$ для граничных точек, реализуем алгоритм **метода Зейделя** в виде

$$U_{ij}^{(k)} = \left(\frac{U_{i,j+1}^{(k)} + U_{i,j+1}^{(k-1)}}{h_1^2} + \frac{U_{i-1,j}^{(k)} + U_{i+1,j}^{(k-1)}}{h_2^2} - F_{ij} \right) / \left(\frac{2}{h_1^2} + \frac{2}{h_2^2} \right) \qquad (3.3)$$

$j = 1,\ldots,N_1 - 1, \quad i = 1,\ldots,N_2 - 1, \quad k = 1, 2, \ldots$

Здесь и в дальнейшем $k = 1, 2, \ldots$ – номер итерации.

Счет ведется до тех пор, пока не будет выполнено условие

$$z_k = \sum_{i=1}^{N_2-1} \sum_{j=1}^{N_1-1} \left| U_{ij}^{(k)} - U_{ij}^{(k-1)} \right| < \varepsilon , \qquad (3.4)$$

где

ε – заданная точность,

z_k – оценка погрешности на k-ом шаге итерации.

Примечание. Сходимость метода Зейделя для задач такого типа доказывается в более подробных курсах вычислительной математики. Она следует из, так называемой, положительной определенности оператора (матрицы) краевой задачи. Отметим, что в данном случае диагонального преобладания нет (и не требуется).

3.3. Решение задачи Дирихле методом конечных разностей в MATLAB

Задача

Решить задачу Дирихле для уравнения Пуассона в прямоугольной области (рис.3.3) используя метод конечных разностей

Для сооружения, имеющего в плане форму квадрата со стороной dl1=dl2=1, найти форму кровли **U(x,y)** в двух вариантах:

- подвесная с удельной нагрузкой **c=8(g+s);**
- надувная с избыточным давлением - **c.**

Сверху противоположные стены попарно ограничены кривыми:

U= 4*g/dl2*(dl2-y)*y

U=4*s/dl1*(dl1-x)*x

Принять модель $\Delta U = \pm c$ знак плюс для подвесной и минус для надувной кровли соответственно.

Краевыми условиями служат верхние границы стен.

Для g=3 и s=12

Исходные данные

$F(x,y) = c$, где $c = 120$;

$$U_{\Gamma} = \varphi(x,y) = \begin{cases} \varphi_1(y) = \varphi_2(y) = 12\,y(\ell_2 - y), \ (x = 0 \vee x = \ell_1) \wedge 0 \le y \le 1 \\ \varphi_3(x) = \varphi_4(x) = 48\,x(\ell_1 - x), \ 0 \le x \le \ell_1 \wedge (y = 0 \vee y = \ell_2) \end{cases};$$

$\ell_1 = \ell_2 = 1;$

Текст М-файла для этой задачи приведен ниже:

```
%Задание количеств точек разбиения
n1=input('Введите  n1= ');
n2=input('Введите  n2= ');
%Задание габаритов области
dl1= input('Введите  dl1= ');
dl2= input('Введите  dl2= ');
h1=dl1/n1; h2=dl2/n2;
c=120;
%Задание граничных условий
for i=1:n2+1
        y=h2*(i-1);
        for j=1:n1+1
            x=h1*(j-1);
    %   u(i,j)=0.;
            if(j==1 || j==n1+1)
                u(i,j)=12/dl2*y*(dl2-y);
            end
            if (i==1||i==n2+1)
                u(i,j)=48/dl1*x*(dl1-x);
            end
        end
end

k=0;
id=0;
while (id==0)
            z=0;
        for i=2:n2
```

```
            for j=2:n1
                r=u(i,j);
                 u(i,j)=((u(i-1,j)+u(i+1,j))/h2^2+...
                 (u(i,j-1)+u(i,j+1))/h1^2-c)/ ...
                 (2/h1.^2+2/h2.^2);
                z=z+abs(u(i,j)-r);
            end
        end
        k=k+1;
        if (z<0.001 && k<100)
            id=1;
        end
end
fprintf('Количество итераций k=%4d \n',k);
disp('Решение u');
for i=1:n2+1
        fprintf('%6.2f', u(i,:));
        fprintf('\n');
end
%Вывод графических изображений
i=1:n2+1;
j=1:n1+1;
[I,J]=meshgrid(i,j);
% подвесная кровля

colormap(copper)
surfl(I,J,u')
shading interp

% надувная кровля
figure;
colormap(copper)
surfl(I,J,-u')
shading interp
```

Результаты приведены ниже, изображения на рисунках 3.5 и 3.6

```
Введите  n1= 8
Введите  n2= 6
Введите  dl1= 1
Введите  dl2= 1
Количество итераций k=  40
Решение u
  0.00  5.25  9.00 11.25 12.00 11.25  9.00  5.25  0.00
  1.67  1.73  2.46  3.11  3.36  3.11  2.46  1.73  1.67
```

```
2.67   0.36 -0.63 -0.97 -1.05 -0.97 -0.63   0.36   2.67
3.00   0.00 -1.53 -2.20 -2.39 -2.20 -1.53   0.00   3.00
2.67   0.36 -0.63 -0.97 -1.05 -0.97 -0.63   0.36   2.67
1.67   1.73  2.46  3.11  3.36  3.11  2.46   1.73   1.67
0.00   5.25  9.00 11.25 12.00 11.25  9.00   5.25   0.00
```

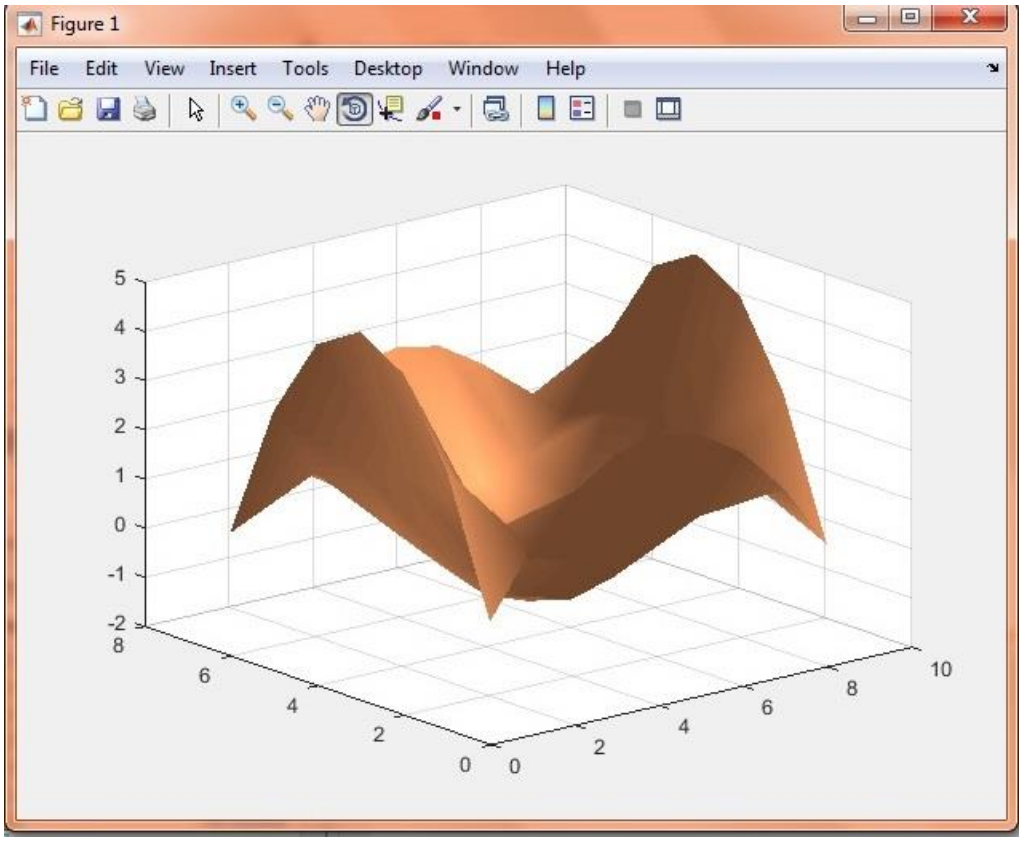

Рис. 3.5 Подвесная кровля

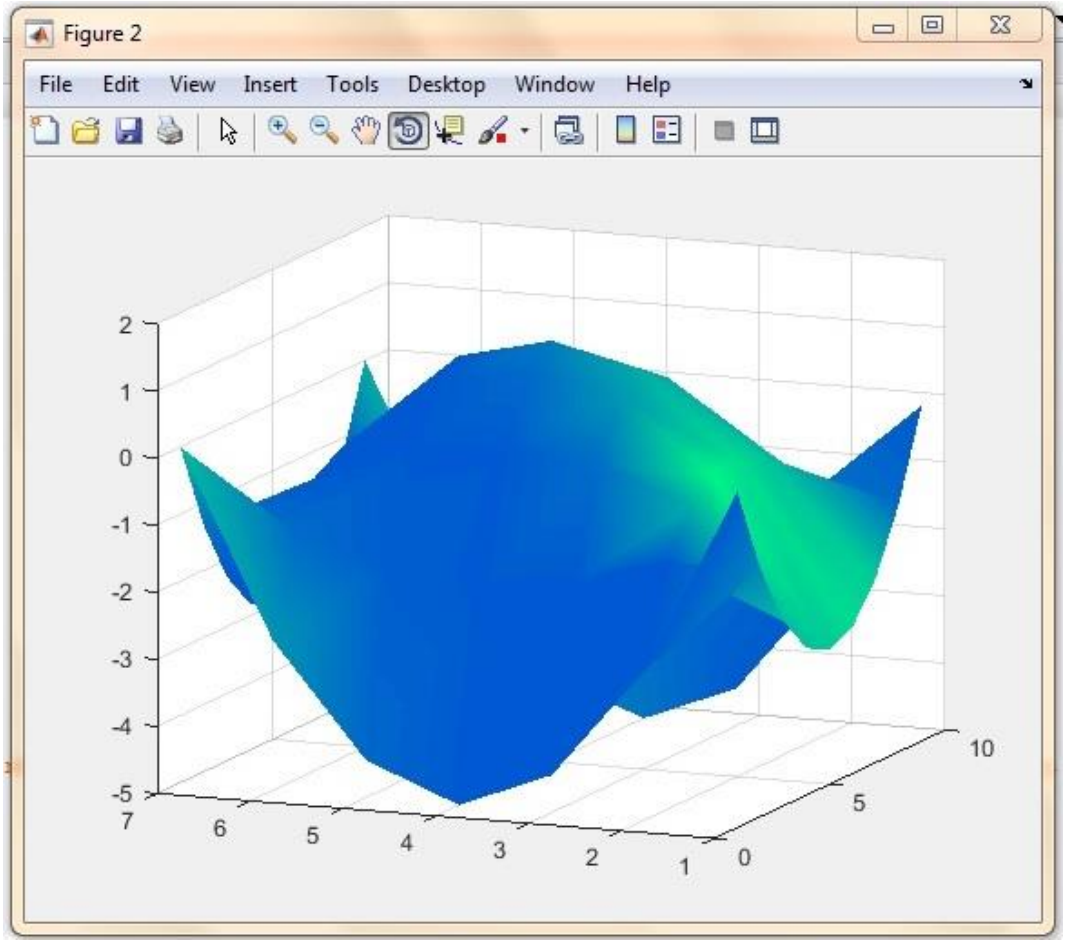

Рис.3.6 Надувная кровля

Лекция 4

ЗАДАЧА КОШИ (задача с начальными условиями)

Введение

Технические задачи, связанные с дифференциальными уравнениями, включают в себя, как правило, описание области (геометрии конструкции, временной промежуток и т.д.) При этом все дополнительные граничные условия могут задаваться только в начале области (начальный момент времени или основание конструкции). Такая задача в отличие от краевой задачи называется задачей **Коши** (задачей с начальными условиями). В частности, это задачи, связанные с временными процессами (распределение температуры, колебания конструкции и т.д.).

Пример 1. Пусть задано дифференциальное уравнение 1-го порядка

$$y' = F(x, y) \qquad (4.1)$$

Будем рассматривать его в области $x > 0$. Функция $F(x, y)$ задана. Неизвестны функции $y(x)$ и $y'(x)$.

Как правило, любое дифференциальное уравнение имеет множество решений. Поэтому для единственности решения требуется дополнительное условие, например, вида (рис. 4.1):

$y(0) = y_0 = g$ – **начальное условие** (задано).

Таким образом, поставленная задача называется *задачей Коши* и имеет единственное решение.

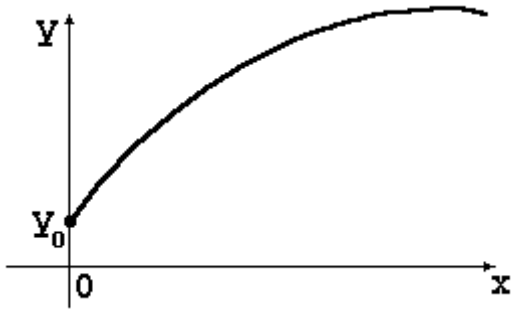

Рис.4.1

Пример 2. Приведем пример задачи Коши для уравнения 2-го порядка:

$$\left.\begin{cases} Ry'' + cy = f, \quad x > 0, \\ y(0) = y_0 = g_1 \\ y'(0) = y_0' = g_2 \end{cases}\right\} - \textit{начальные} \quad \textit{условия}. \tag{4.2}$$

Неизвестной является функция $y(x)$.

То есть, для функции $y(x)$ заданы начальное значение y_0 и начальный угол наклона y_0' (рис. 4.2).

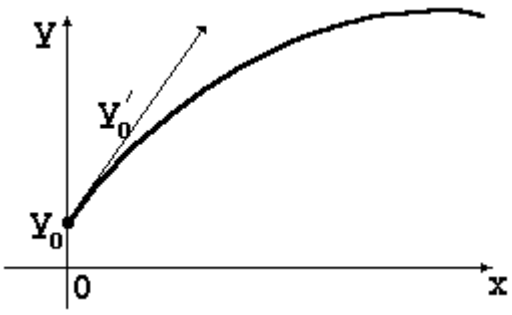

Рис. 4.2

Сведение задачи (4.2) к системе первого порядка

Введем дополнительную функцию

$z(x) = y'(x)$.

Подставляя ее в (4.2) получим:

$$\begin{cases} \begin{cases} Rz' + cy = f \\ y' - z = 0 \end{cases}, \quad x > 0 \\ \left.\begin{matrix} y(0) = y_0 = g_1 \\ z(0) = z_0 = g_2 \end{matrix}\right\} - \textit{начальные} \quad \textit{условия}, \end{cases} \tag{4.3}$$

где $z_0 = y'_0$.

4.1. Метод Эйлера

В общем случае задача Коши аналитического решения не имеет, поэтому применяются численные методы, простейшим из которых является **метод Эйлера**. Воспользуемся для решения задачи Коши методом конечных разностей. Для этого зададимся достаточно малым шагом h. Будем рассматривать решение в точках с координатами $x_i : x_i = i \cdot h$, $i = 0,1,2, \ldots$.

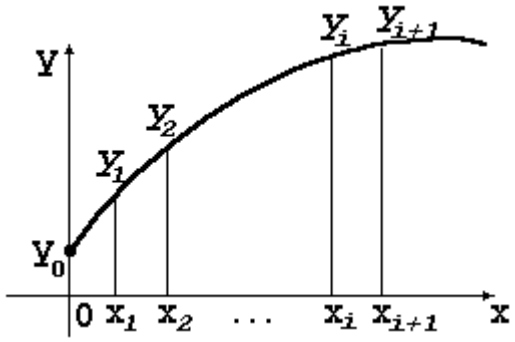

Рис.4.3

$h_i = x_{i+1} - x_i$

Обозначим $y_i = y(x_i)$, $f_i = f(x_i)$, $y'_i = y'(x_i)$

и т.д.

Заменим производную конечно-разностным отношением: $y_i' \approx \frac{y_{i+1} - y_i}{h}$.

Получим конечно-разностные уравнения вида:

для **примера 1**:

$$\begin{cases} y_0 = g, \\ \frac{y_{i+1} - y_i}{h} = F(x_i, y_i), \quad i = 0,1,2,\ldots \end{cases}$$

или

$$\begin{cases} y_0 = g, \\ y_{i+1} = y_i + h \cdot F(x_i, y_i), \quad i = 0,1,2,\ldots . \end{cases} \tag{4.4}$$

Аналогично для **примера 2** (4.3) получим:

$$\begin{cases} y_0 = g_1 \\ z_0 = g_2 \end{cases} \text{ – начальные значения}$$

$$\begin{cases} R_i \frac{z_{i+1} - z_i}{h} + cy_i = f_i \\ \frac{y_{i+1} - y_i}{h} = z_i \end{cases}, \; i = 0,1,2, \ldots$$

или

$$\begin{cases} y_0 = g_1 \\ z_0 = g_2 \end{cases} \text{ – начальные значения}$$

$$\begin{cases} z_{i+1} = z_i - h \cdot (cy_i - f_i)/R_i \\ y_{i+1} = y_i + h \cdot z_i \end{cases}, \; i = 0,1,2, \ldots \tag{4.5}$$

Вместо последней формулы можно записать более точную

$$y_{i+1} = y_i + h \cdot \frac{z_i + z_{i+1}}{2}.$$

Таким образом, в каждой точке разбиения количество неизвестных совпадает с количеством уравнений. Приведенные формулы (4.4) и (4.5)

являются рекуррентными формулами вычислений: зная значения x_i, y_i, z_i, вычисляем по ним следующие значения $x_{i+1}, y_{i+1}, z_{i+1}$ и т.д. Пользуясь таким алгоритмом, последовательно получим решение для любого количества точек разбиения. Вышеизложенный конечно-разностный подход для решения задачи Коши называется **методом Эйлера**.

Примечание. В сложных практических случаях применяются различные модифицированные алгоритмы, связанные с уточнением шага разбиения на каждом шаге пересчета. Среди такого рода модификаций наиболее употребляемыми являются методы типа Рунге-Кутта, излагаемые в специальных пособиях.

4.2 Задача об изгибе консоли

Рассмотрим задачу об изгибе консоли, закрепленной в левой части.

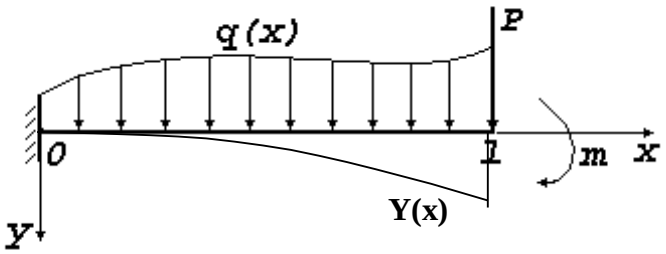

Рис. 4.4

В курсе «Сопротивление материалов» приводится нелинейное дифференциальное уравнение прогиба оси консоли и начальные условия

$$\begin{cases} y''(x) = \dfrac{M(x)}{EJ(x)}\sqrt{[1+(y'(x))^2]^3}, \\ y(0) = y'(0) = 0 - \quad \textit{начальные} \quad \textit{условия}\ , \end{cases} \tag{4.6}$$

где $M(x)$ – момент, возникающий в консоли от приложенных нагрузок

$$M(x) = -m - P(1-x) - \int_x^{\ell} (x-\xi) q(\xi) d\xi \ .$$

В частности, при постоянной нагрузке $q(x)$: $q(x) = q = const$, выражение для $M(x)$ может быть записано проще (в этом случае легко вычисляется интеграл в правой части)

$$M(x) = -m - P(\ell - x) - q\frac{(\ell - x)^2}{2}$$

Обозначив $z(x) = y'(x)$, перейдем к системе двух уравнений 1-го порядка

$$\begin{cases} z'(x) = \dfrac{M(x)}{EJ(x)} \sqrt{[1 + z^2(x)]^3} \\ y'(x) = z(x) \end{cases},$$

$$\begin{cases} z(0) = 0 \\ y(0) = 0 \end{cases} \text{ – начальные условия.}$$

Зададим шаг разбиения h и будем рассматривать решение в точках с координатами x_i: $x_i = i \cdot h$, $i = 0,1,2, \ldots$. Получим следующие разностные уравнения

$$\begin{cases} z_0 = 0, \\ y_0 = 0, \\ \dfrac{z_{i+1} - z_i}{h} = \dfrac{M_i}{EJ_i} \sqrt{(1 + z_i^2)^3}, \quad i = 0,1,2, \ldots . \\ \dfrac{y_{i+1} - y_i}{h} = z_i, \end{cases}$$

Этим уравнениям соответствуют рекуррентные формулы пересчета по методу Эйлера

$$z_0 = y_0 = 0$$

$$\begin{cases} z_{i+1} = z_i + h \cdot \dfrac{M_i}{EJ_i} \sqrt{(1 + z_i^2)^3} \\ y_{i+1} = y_i + \dfrac{h}{2} \cdot (z_i + z_{i+1}) \end{cases}, \qquad i = 0,1,2, \ldots, N-1$$

Для расчета возьмем следующие соотношения:

$$M(x)=\frac{1}{\sqrt{[1+(cx)^2]^3}},\qquad EJ=c^{-1},\ c=0{,}3,\ l=1,.$$

Текст М-функции приведен ниже.

```
function zcoshi

dl=input('введите dl');
N=input('введите N ');
x0=input('введите x0 ');
y0=input('введите y0 ');
z0=input('введите z0 ');

dl,N,x0,y0,z0
h=dl/N
c=0.3;

disp('       x          y          z');
d=zeros(N+1,3); d1=zeros(N+1,1); d2=zeros(N+1,1); d3=zeros(N+1,1);
for i=1:N+1
 d1(i)=x0;
 d2(i)=y0;
 d3(i)=z0;
 z=z0+h*f(x0,z0,c);
 y=y0+h*(z0+z)/2;
 x0=x0+h;
 z0=z;
 y0=y;
end
d=[d1 d2 d3];
disp(d);

function y=f(x,z,c)
y=c*M(x,c)*sqrt((1+z^2)^3);

function u=M(x,c)
u=1/sqrt((1+(c*x)^2)^3);
```

Результаты приведены ниже

```
введите dl  1
введите N  10
введите x0  0
введите y0  0
введите z0  0

dl =

     1

N =
```

```
    10

x0 =

     0

y0 =

     0

z0 =

     0

h =

    0.1000

       x          y          z
         0          0          0
    0.1000    0.0015    0.0300
    0.2000    0.0060    0.0600
    0.3000    0.0135    0.0900
    0.4000    0.0240    0.1200
    0.5000    0.0375    0.1500
    0.6000    0.0540    0.1800
    0.7000    0.0735    0.2100
    0.8000    0.0960    0.2400
    0.9000    0.1215    0.2700
    1.0000    0.1500    0.3000
```

4.3 Оценка погрешности метода Эйлера в MATLAB

В качестве примера рассмотрим поведение оценки погрешности на примере численного решения уравнения вида:

$$u' = xu, \quad u(0) = 1, \quad x \in [0,1].$$

Это уравнение имеет точное решение

$$u = e^{\frac{x^2}{2}},$$

которое сравним с приближенным решением y_n, определяемым на равномерной сетке $x_n = (n-1)/h, \quad n = 1,2,...,n,$ где $h = 1/(N-1)$.

Изучим зависимость величины константы в точке x=1

$$M(1) = \frac{1}{h} | y_N - e^{1/2} |$$

от шага сетки.

Ниже приведен текст М-файла

```
%Программа изучения погрешности численного решения
%уравнения u'=f(x,u) методом Эйлера

clear
%Задаем набор сеток
Mesh=10:10:10000;
%Организуем цикл расчета методом Эйлера на разных сетках
for k=1:length(Mesh)
%Определяем параметры сетки
        N=Mesh(k); h=1.0/(N-1);
%Задаем начальные условия
        y(1)=1;
%Алгоритм метода Эйлера
        for n=1:(N-1)
            y(n+1)=y(n)+(n-1)*h^2*y(n);
        end
%Вычисляем M(1) в оценке погрешности численного решения
    M(k)=abs(y(N)-exp(0.5))/h;
step(k)=h;
end
Рисуем зависимость M(1) от шага разностной  сетки
semilogx(step,M);
```

На рисунке 4.5 видно, что при стремлении шага сетки к нулю величина M(1) ведет себя как константа, т.е. не зависит от шага сетки.

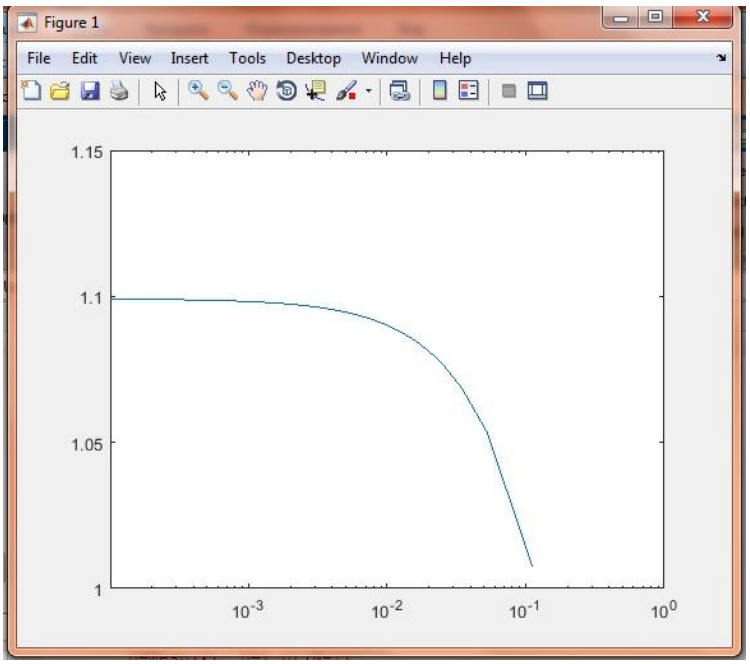

Рис.4.5 Изучение погрешности метода Эйлера.

4.4 Понятие об устойчивости разностной схемы

Рассмотрим простейшую задачу Коши вида

$$\begin{cases} y' = -\lambda y, & x > 0 \\ y(0) = y_0 \end{cases}, \quad \text{где } \lambda > 0 .$$

В этом случае известно аналитическое решение

$$y = y_0 e^{-\lambda x},$$

из которого видно, что

$$y(x) \xrightarrow[x\to\infty]{} 0 .$$

Разностная формула метода Эйлера имеет вид

$$y_{i+1} = y_i - h\lambda y_i = (1 - h\lambda) y_i .$$

Откуда следует, что

$$y_i = (1 - h\lambda)^i y_0 .$$

Обозначим

$$q = 1 - h\lambda$$

Тогда

$y_i = q^i y_0$

При этом, очевидно, что

$y_i \xrightarrow[i\to\infty]{} 0$

только при $|q| < 1$, что соответствует соотношениям

$-1 < 1 - h\lambda < 1$ или $0 < h\lambda < 2$

откуда следуют ограничения для величины h:

$$0 < h < \frac{2}{\lambda}.$$

Последнее неравенство называется условием **устойчивости** счета.

Абсолютно устойчивая схема

Существуют разностные схемы, обеспечивающие устойчивость счета при любом шаге h. В частности, для нашего примера абсолютно устойчивой будет схема

$$\frac{y_{i+1} - y_i}{h} = -\lambda \frac{y_{i+1} + y_i}{2},$$

или

$$y_{i+1} = \frac{1 - \frac{\lambda h}{2}}{1 + \frac{\lambda h}{2}} y_i$$

В этом случае

$$q = \frac{1 - \frac{\lambda h}{2}}{1 + \frac{\lambda h}{2}}.$$

Очевидно, что

$$|q| < 1 \quad \text{и} \quad y_i \xrightarrow[i\to\infty]{} 0,$$

т.е. такая схема устойчива при любом h.

Замечание. В общем случае (при произвольной функции $F(x,y)$) абсолютно устойчивой разностной схеме соответствует разностное уравнение вида

$$\frac{y_{i+1}-y_i}{h}=F(x_i,\frac{y_{i+1}+y_i}{2})\ .$$

Из такого уравнения не всегда удается выделить y_{i+1} (для этого может потребоваться явное решение нелинейного уравнения).

4.5 Задача Коши для обыкновенных дифференциальных уравнений. Динамическая система

Исследуем поведение пружинного маятника с колеблющееся массой **m**, упругостью пружины **k** и коэффициентом сопротивления (вязкого трения) **r** в интервале от 0 до 10 секунд с шагом 0.1 сек при следующих значениях параметров:

m=1, k=1, r= 0.45

Начальная скорость в положении равновесия равна -0.45

Построим зависимость положения **x(t)** и скорости x'(t) от времени, а также фазовую траекторию $\dot{x}$ **(x)** и интегральную кривую **x(t),** $\dot{x}$ **(t).**

Используем модификацию метода Эйлера с вычислением производных на половинном шаге.

Математической моделью данной физической системы является уравнение

$$\ddot{x}+r\dot{x}+kx=0 \qquad (4.7)$$

с начальными условиями

$$x(0)=0$$

$$\dot{x}(0)=-45$$

Приведем уравнение (4.7) к нормальной системе путем замены

$$x_1=x$$

$$x_2=\dot{x}$$

Получим систему

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -kx_1 - rx_2 \end{cases}$$

с начальными условиями

$$x_1(0)=0$$

$$x_2(0)= -45$$

Создаем М-файл

```
function bob
tau=0.1;
t=0:tau:10;
[n1,n2]=size(t);
x=zeros(2,n2);
x(2,1)=-45;
xx=x;
for i=1:n2-1
        xx(:,i+1)=x(:,i)+0.5*tau*f(x(:,i));
        x(:,i+1)=x(:,i)+tau*f(xx(:,i+1));
end
subplot(2,2,1); plot(t,x(1,:)),xlabel('t'),ylabel('x1'),
grid on
subplot(2,2,2); plot(t,x(2,:)),xlabel('t'),ylabel('x2'),
grid on
subplot(2,2,3);
plot(x(1,:),x(2,:)),xlabel('x1'),ylabel('x2'), grid on
subplot(2,2,4);
plot3(t,x(1,:),x(2,:)),xlabel('t'),ylabel('x1'),
zlabel('x2'), grid on

function u=f(x)
u=[x(2); -x(1)-0.45*x(2)];

end
end
```

Результат представлен на рисунке 4.6

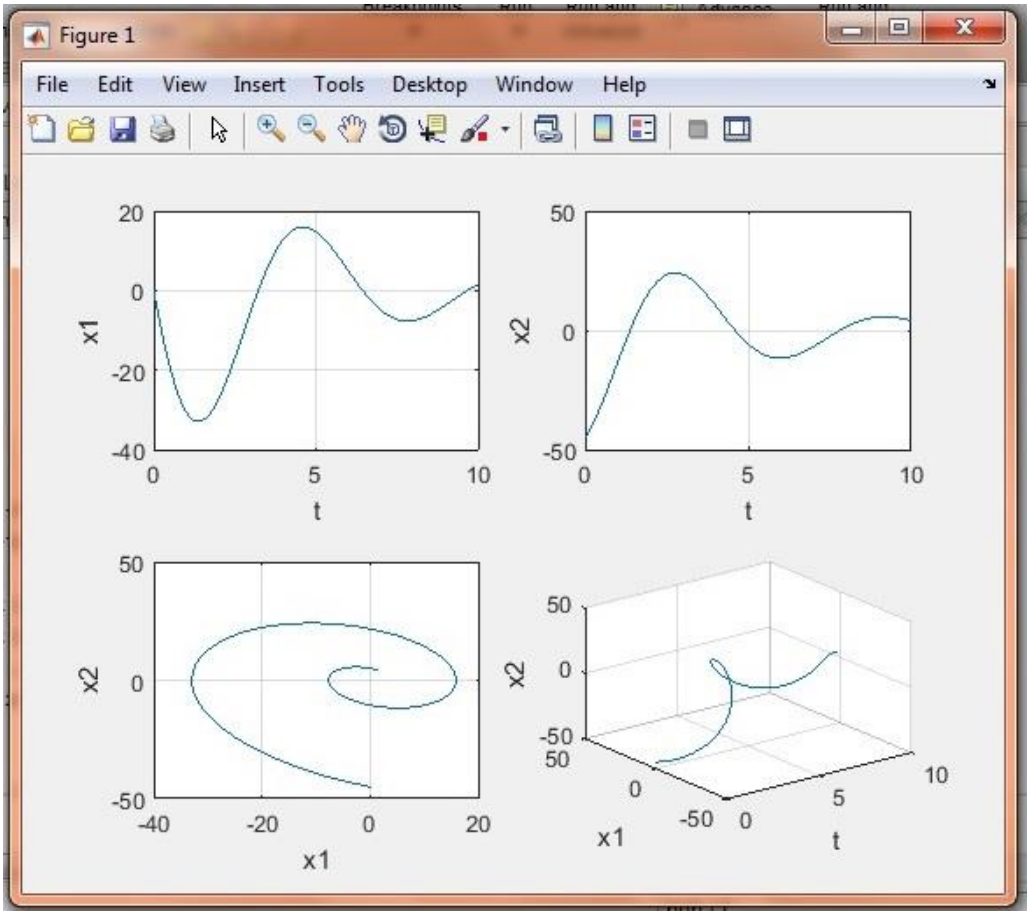

Рис. 4.6 Задача Коши для мятника

4.6 Средства MATLAB для решения задачи Коши для обыкновенных дифференциальных уравнений

В системе MATLAB имеется целый ряд встроенных функций, предназначенных для решения задачи Коши для обыкновенных дифференциальных уравнений. Краткая характеристика этих функций, называемых солверами (от английского слова solver, переводимого на русский язык как "решатель") приведена в таблице 4.1 В них реализованы различные методы решения задачи Коши для дифференциальных уравнений, поэтому выбор той или иной функции зависит от характеристик решаемой задачи.

Таблица 4.1

Краткая характеристика солверов

Название функции	Характеристика функции

ode45	реализует одношаговые явные методы Рунге-Кутта 4-го и 5-го порядка. Это классический метод, рекомендуемый для начальной пробы решения. Во многих случаях он дает хорошие результаты. Применяют часто эту функцию, если характеристики задачи неизвестны.
ode23	реализует одношаговые явные методы Рунге-Кутта 2-го и 4-го порядка. При умеренной жесткости системы ОДУ и низких требованиях к точности этот метод может дать выигрыш в скорости решения.
ode113	реализует многошаговый метод Адамса-Башворта-Мултона переменного порядка. Это адаптивный метод, который может обеспечить высокую точность решения. Рекомендуется для решения нежестких задач с высокой точностью.
ode23tb	реализует неявный метод Рунге-Кутта в начале решения и метод, использующий формулы обратного дифференцирования 2-го порядка в последующем. Несмотря на относительно небольшую точность, данный метод может оказаться более эффективным, чем, например, **ode15s**.
ode15s	реализует многошаговый метод переменного порядка (от 1 до 5, по умолчанию 5), использующий формулы численного "дифференцирования назад". Это адаптивный метод, его стоит применять, если решатель ode45 не обеспечивает решения и система дифференциальных уравнений жесткая.
ode23s	реализует одношаговый метод, использующий модифицированную формулу Розенброка 2-го порядка.

	Может обеспечить высокую скорость вычислений при низкой точности решения жесткой системы дифференциальных уравнений
ode23t	реализует метод трапеций с интерполяцией. Этот метод дает хорошие результаты при решении задач, описывающих колебательные системы с почти гармоническим выходным сигналом;
bvp4c	служит для решения проблемы граничных значений систем дифференциальных уравнений вида y'= f(t,y), F(y(a), y(b), p) = 0 (краевая задача);
pdepe	нужен для решения систем параболических и эллиптических дифференциальных уравнений в частных производных, введен в ядро системы для поддержки новых графических функций Open GL, пакет расширения Partial Differential Equations Toolbox содержит более мощные средства.

Название этих функций начинается на ode —ordinary differential equetions (обыкновенные дифференциальные уравнения).

Все решатели, которые представлены в таблице 4.1, могут решать системы уравнений первого порядка

$$y' = F(x, y),$$

причем для решения жестких систем уравнений рекомендуется использовать лишь специальные функции **ode15s, ode23s, ode23t, ode23tb.**

Решатели **ode15s** и **ode23t** способны найти корни дифференциально-алгебраических уравнений вида

$$My' = F(x, y),$$

где M называется матрицей массы.

Все решатели, за исключением **ode23s,** который требует постоянства матрицы массы, могут решать матричного уравнения вида

M(x,y) $y' = F$(x, y).

Заметим, что **ode23tb**, **ode23s** служат для решения жестких дифференциальных уравнений, **ode15s** -жестких дифференциальных и дифференциально-алгебраических уравнений, **ode23t** -умеренно жестких дифференциальных и дифференциально-алгебраических уравнений.

Использование решателей систем ОДУ

В описанных далее функциях для решения систем дифференциальных уравнений приняты следующие обозначения и правила:

- **options** — аргумент, создаваемый функцией **odeset** (еще одна функция — **odeget** или **bvpget** (только для **bvp4c**)— позволяет вывести параметры, установленные по умолчанию или с помощью функции **odeset** или **bvpset** соответственно);
- **xspan** — вектор, определяющий интервал интегрирования [x0 xfinal]. Для получения решений в конкретных точках разбиения x0, x1,..., xfinal (расположенные в порядке уменьшения или увеличения) нужно использовать **xspan = [x0 xl ... xfinal]**;
- **y0** — вектор начальных условий;
- **p1, p2,…** — произвольные параметры, передаваемые в функцию **F**;
- **X, Y** — матрица решений Y, где каждая строка соответствует определенной координате точки разбиения, возвращенном в векторе-столбце **X**.

Перейдем к описанию функций для решения систем дифференциальных уравнений:

- **[X,Y] = solver(@F,xspan,y0)** — где вместо solver подставляем имя конкретного решателя — решается система дифференциальных уравнений вида $y'=F$(x,y) на интервале **xspan** с начальными условиями **y0**, **@F** — дескриптор ODE-функции (также можно задать функцию вида 'F', т.е. указать имя файл-функции в одинарных кавычках) или, иными словами, указатель на функцию, которая вычисляет правые части системы

дифференциальных уравнений. Каждая строка в массиве решений **Y** соответствует значениям координат точек разбиения, возвращаемым в векторе-столбце **X**;

- **[X,Y] = solver(@F, xspan, y0. options)** — дает решение, подобное описанному выше, но с параметрами, определяемыми значениями аргумента **options**, созданного функцией **odeset**. Обычно используемые параметры включают допустимое значение относительной погрешности **RelTol** (по умолчанию 1e-3) и вектор допустимых значений абсолютной погрешности **AbsTol** (все компоненты по умолчанию равны 1e-6);
- **[X,Y] = solver(@F,tspan,y0,options,p1,p2,...)** — дает решение, подобное описанному выше, передавая дополнительные параметры **p1, p2**,... в m-файл **F** всякий раз, когда он вызывается. Используйте **options=[]**, если никакие параметры не задаются;
- **[X,XE,YE,IE] = solver(@F,xspan,y0,options)** — в дополнение к описанному решению содержит свойства **Events**, установленные в структуре **options** ссылкой на функции событий. Когда эти функции событий от (x, y) равны нулю, производятся действия в зависимости от значения трех векторов **value**, **isterminal** и **direction** (их величины можно установить в m-файлах функций событий). При этом для i-й функции событий **value(i)** —значение функции, **isterminal(i)** — прекратить интеграцию при достижении функцией нулевого значения; **direction(i) = 0**, если все нули функции событий нужно вычислять (по умолчанию), **direction(i) = +1** — только те нули, где функция событий увеличивается, **direction(i) = -1** — только те нули, где функция событий уменьшается. Выходной аргумент **XE** — вектор-столбец времен, в которые происходят события (**events**), строки **YE** являются соответствующими решениями, а индексы в векторе **IE** определяют, какая из i функций событий (**event**) равна нулю при значении независимой переменной, определенный XE. Когда происходит вызов функции без выходных аргументов, по

умолчанию вызывается выходная функция **odeplot** для построения вычисленного решения.

Параметры интегрирования (**options**) могут быть определены и в m-файле, и в командной строке с помощью команды **odeset**. Если параметр определен в обоих местах, определение в командной строке имеет приоритет.

Решатели используют в списке параметров различные параметры:

- **NormControl** — управление ошибкой в зависимости от нормы вектора решения **on** или **off**. Установите '**on**', чтобы **norm(e)<=max(RelTol*norm(y), AbsTol).** По умолчанию все решатели используют более жесткое управление по каждой из составляющих вектора решения;
- **RelTol** — относительный порог отбора (положительный скаляр). По умолчанию равный 1e-3 (т.е. точность оставляет 0.1%) во всех решателях; оценка ошибки на каждом шаге интеграции **e(i) <= max(RelTol*abs(y(i)), AbsTol(i));**
- **AbsTol** — абсолютная точность (положительный скаляр или вектор, все компоненты которого равны 1e-6). Скаляр вводится для всех составляющих вектора решения, а вектор указывает на компоненты вектора решения. **AbsTol** по умолчанию принимает значение 1e-6 во всех решателях;
- **Refine** - фактор уточнения вывода (положительное целое число) — умножает число точек вывода на этот множитель. По умолчанию всегда равен 1, кроме **ode45**, где он равен четырем. Не применяется, если **xspan>2**;
- **OutputFcn** — дескриптор функция вывода (function) — имеет значение в том случае, если решатель вызывается без явного указания функции

вывода, **OutputFcn** по умолчанию вызывает функцию **odeplot**. Эту установку можно поменять именно здесь;

- **OutputSel** — индексы отбора (вектор целых чисел) Установите компоненты, которые поступают в **OutputFcn**. **OutputSel** по умолчанию выводит все компоненты;
- **Stats** — установка статистики стоимости вычислений. Возможные значения **on** или **off**;
- **Jacobian** — функция матрицы **Якоби** (function constant matrix). Данное свойство следует установить на дескриптор функции **FJac** (если **FJac** возвращает **dF/dy**) или на имя постоянной матрицы **dF/dy**;
- **Jpattern** — график разреженности матрицы Якоби,т.е. имя разреженной матрицы. Матрица **S** с **S(i,j) = 1**, если i-ая составляющая **F(x, y)** зависит от **j-ой** составляющей величины **y**, и **S(i,j) =** 0 в противоположном случае;
- **Vectorized** — векторизованная функция правых частей, возможные значения которой **on** или **off**. Устанавливается на '**on**', если функция правых частей **F(x,y)** такая, что **F(x,[yl y2...])** возвращает вектор **[F(x, yl) F(t, y2) ...];**
- **Events** —(function) — ввод дескрипторов функций событий, содержащих собственно функцию в векторе **value**, а векторы **isterminal** и **direction** (см выше);
- **Mass** — матрица массы (constant matrix function), т.е. имя постоянной матрицы для задач **My' = *F*(x, y)** или дескриптор функции, описывающей матрицу массы для задач с переменной **M** вида **M(x,y) *y′* = *F*(x, y)**;
- **MstateDependence** — зависимость матрицы массы от y, возможные значения: **none**, weak или **strong**. Если зависимость отсутствует, то для уравнений вида **My' = *F*(x, y)** или **M(x)*y' = *F*(x, y)** устанавливается '**none**'. '**weak**' (зависимость слабая**)** устанавливается для уравнений **M(x,y) *y′* = *F*(x, y),** но '**weak**' приводит к неявным алгоритмам решения,

использующим аппроксимации при решении алгебраических уравнений. **strong** -(зависимость сильная) задается для уравнений **M(x,y)*y′*=*F*(x,y)**;

- **MassSingular** — матрица массы M сингулярная, возможные значения **yes (да), no (нет)** или **maybe (может быть**);
- **MvPatter**n — разреженность (**dMv/dy**), график разреженности (см функцию **spy**) - требуется ввести имя разреженной матрицы **S** с **S(i,j) = 1** для любого **k**, где **(i, k)** элемент матрицы массы **M(x, y)** зависит от проекции **j-ой** переменной y, и **S(i,j) =** 0 в противном случае;
- **InitialStep** — предлагаемый начальный размер шага, по умолчанию каждый решатель определяет его автоматически по своему алгоритму;
- **InitialSIope** — вектор начального уклона **yp0**, где соответствующий начальный уклон определяется формулой **yp0 = F(x0,y0)/M(x0, y0**);
- **MaxStep** — максимальный шаг, по умолчанию во всех решателях равен одной десятой интервала **xspan**;
- **BDF (Backward Differentiation Formulas)** [on | {off}] — указывает, нужно ли использовать формулы обратного дифференцирования (методы **Gear**) вместо формул численного дифференцирования, используемых в **ode15s** по умолчанию. Возможные значения **on** или **off**;
- **MaxOrder** - максимальный порядок **ode15s.** Возможные значения: [1 | 2 | 3 4 | 5].

Решатели используют в списке различные параметры. В приведенной ниже таблице они даны для решателей обычных (в том числе жестких) дифференциальных уравнений.

Таблица 4.2

Параметры решателей, используемые в списке опций.

Параметры	**Ode45**	**Ode23**	**Ode11s**	**Ode15s**	**ode23s**
RelTol,AbsTol	+	+	+	+	+
OutputFcaOutputSel, Refine, Stats	+	+	+	+	+

Events	+	+	+	+	+
MaxStep, InitlalStep	+	+	+	+	+
Jacobian, Jpattern, Vectorized	-	-	-	+	+
Mass	-	-	-	+	+
MassConstant	-	-	-	+	-
MaxOrder, BOF	-	-	-	+	-

Решатель **bvp4c**, который мы рассмотрели ранее, имеет очень небольшое число параметров, но при его использовании можно вводить не только матрицу Якоби интегрируемой функции, но и матрицу Якоби, содержащую частные производные функции граничных условий по границам интервала и по неизвестным параметрам.

Для пояснения понятия жесткости системы дифференциальных уравнений рассмотрим на ставшем классическом примере — решение уравнения **Ван-дер-Поля**. Уравнение Ван-дер-Поля описывает релаксационные колебания в электронных устройствах. Это уравнение является жестким. Покажем это.

Уравнение Ван-дер-Поля

$$u''+u-k(1-u^2)u'=0$$

перепишем в векторной форме:

$$\frac{d\bar{u}}{dt}=\begin{Vmatrix} 0 & 1 \\ -1 & k(1-u_1^2) \end{Vmatrix}\bar{u} \qquad (4.8)$$

,

где $\bar{u}=\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \quad u_1=u, \quad u_2=u'$

Величину $(1-u_1^2)=b$ можно считать порядка единицы, т.е.|b|~1. Находя собственные значения матрицы (4.8) при k>>1, имеем λ_1=bk, λ_2=1/(bk).

Число жесткости s определяется как отношение собственных чисел

$$\mathbf{s=\lambda_1/\lambda_2=b^2k^2}.$$

Для k=1000, доходим, что s~10^6, т.е. система дифференциальных уравнений является жестким.

В системе MATLAB для нахождения числа жесткости используется функция **cond**.

```
>>A=[0 1;-1 1000];
>>cond(A)
ans=
     1e+06
```

Решим нелинейную систему (4.8) с помощью встроенного в MATLAB решателя для жестких систем **ode23s.**

Перед решением нужно записать систему дифференциальных уравнений в виде ode-функции. Для этого создадим М-файл

```
function dydt = vdp(t,u)
% формируем вектор столбец
     dydt = zeros(2,1);
     dydt(1) = u(2);
     dydt(2) = 1000*(1 -u(1)^2)*u(2) -u(1);

end
```

Параметр t будет изменяться в пределах от 0 до 5000, начальные условия вектора u=[0 1].

Тогда решение решателем **ode23s** и сопровождающий его график можно получить, используя следующие команды:

```
>> [T,Y]=ode23s(@vdp,[0 5000],[0 1]);
>> plot(T,Y(:,1),T,Y(:,2), ...
     T(1:(length(T)-1)),diff(T));
>> axis([0,5000,-15,60]);
```

Решение представлено на рисунке на 4.7

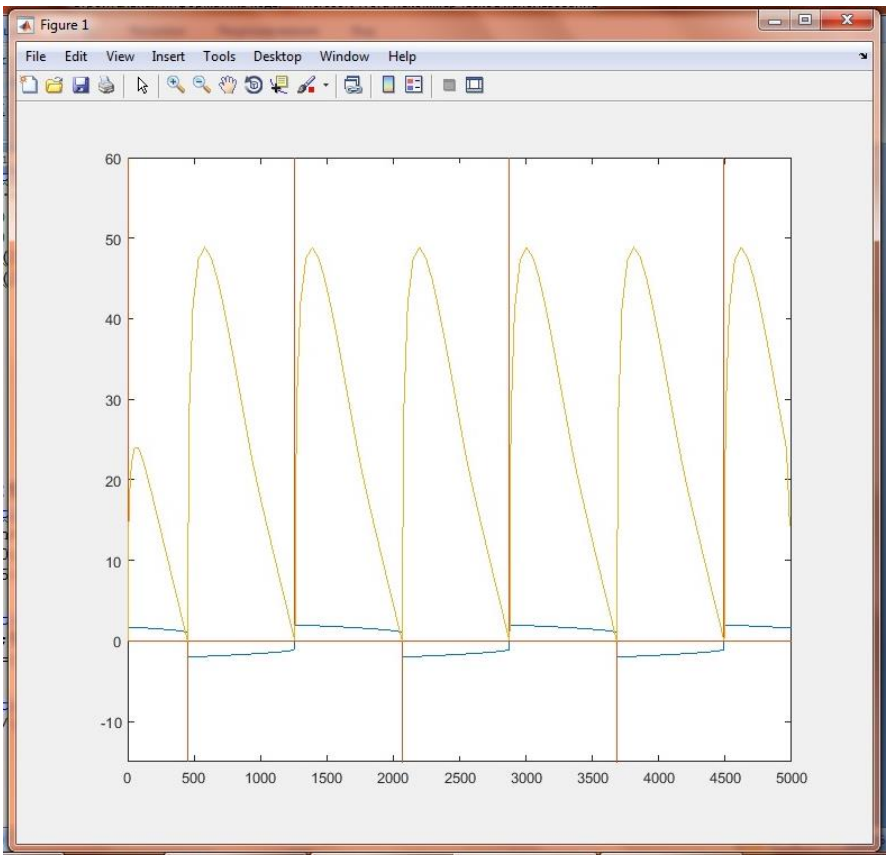

Рис.4.7 Решение уравнения Ван-дер-Поля решателем ode23s

В решателе **ode23s** происходит автоматическое регулирование шага интегрирования. Шаг значительно уменьшается вблизи точек с максимальной производной и, наоборот, вне этих точек существенно увеличивается.

4.7 Решение задачи изгиба консоли функцией MATLAB

Решим рассмотренную выше задачу (4.6) встроенными средствами системы MATLAB — функцией ode45

Для рассмотренной задачи текст М-файла в этом случае будет:

```
function zc
xspan=[0 1];
y0=[0; 0];
ode45(@f,xspan,y0);

function dydx=f(x,y)
c=1.;
dydx=[y(2);M(c*x)*c*(sqrt(1+y(2)^2))^3];
```

```
function u=M(x)
u=-1/(sqrt(1+x^2))^3;
end
end
end
```

Получаем графики прогиба (верхняя кривая на рисунке 4.8) и угла поворота

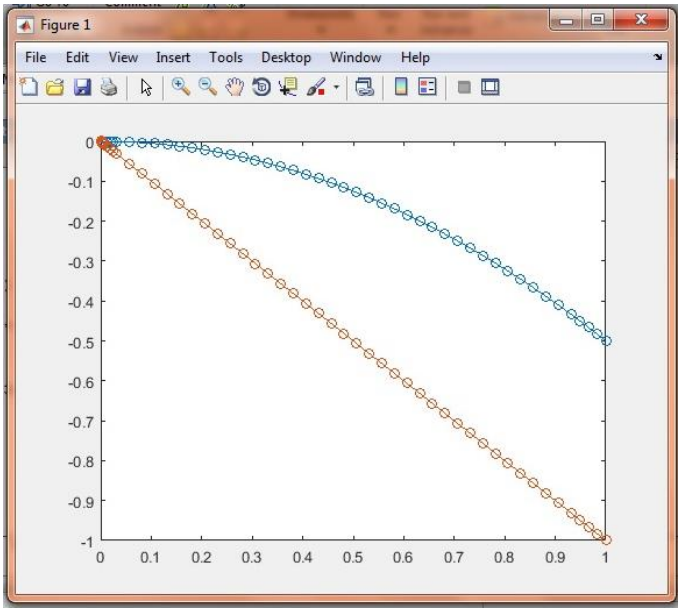

Рис.4.8 Графики прогиба и угла поворота

Лекция 5

ЗАДАЧА ТЕПЛОПРОВОДНОСТИ

Введение

В задачах расчета сооружений большое значение имеет задача расчета температуры в конструкции, так как она определяет необходимые условия функционирования сооружений, а также термонапряженное состояние конструкции.

Пример. Определить распределение температуры по толщине стены для заданного промежутка времени.

Будем считать, что распределение температуры в стене зависит только от расстояния точки от наружной части стены и времени (это значит, что стена имеет достаточно большие размеры). Полагаем, также, что температура на улице и в помещении в любой момент времени известна. Кроме того, будем считать, что в исходный момент времени распределение температуры по толщине стены также известно. Это может быть известно из какого-нибудь предварительного расчета, например, если исходному моменту предшествовал температурный период стабильности.

Примечание 1. В действительности по прошествии достаточного времени распределение температуры мало зависит от начального распределения.

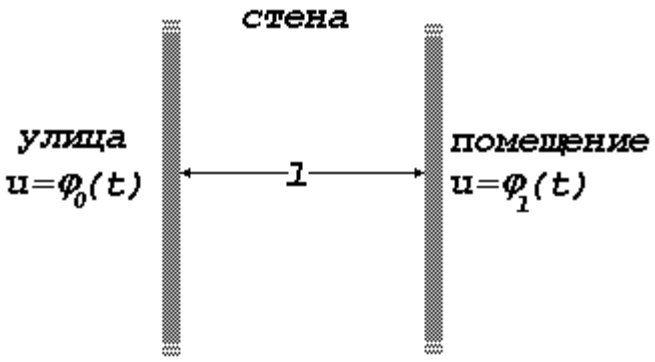

Рис. 5.1

5.1 Математическая формулировка задачи.

$$\begin{cases} \dfrac{\partial u}{\partial t} = \alpha \dfrac{\partial^2 u}{\partial x^2} + f(x,t) \quad - \textit{уравнение теплопроводности}, \\ 0 < x < \ell, \quad t > 0 \quad - \textit{пространственно} - \textit{временная область}, \\ \left.\begin{array}{l} u(0,t) = \varphi_0(t) \\ u(\ell,t) = \varphi_\ell(t) \end{array}\right\}, \quad t \geq 0 \quad - \textit{краевые условия}, \\ u(x,0) = \psi(x), \quad 0 \leq x \leq \ell \quad - \quad \textit{начальные условия}. \end{cases} \tag{5.1}$$

где

x – координата по длине: $0 \leq x \leq \ell$;

t – координата по времени: $t \geq 0$;

$u(x,t)$ – значение температуры в точке x во время t;

α – коэффициент температуропроводности;

$f(x,t)$ – функция источника тепла;

ℓ – толщина стены.

Задача (5.1) определена в пространственно-временной области Ω:

$\Omega = \{x,t : 0 < x < \ell, \ t \geq 0\}$

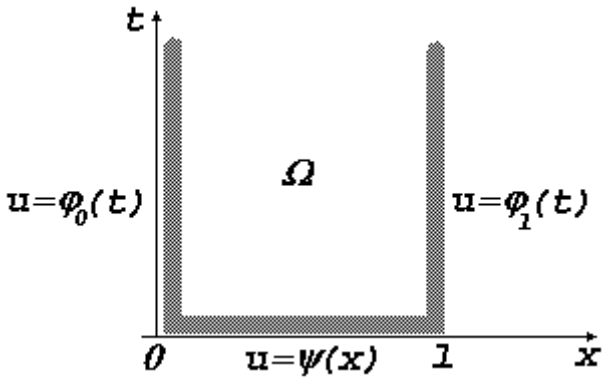

Рис. 5.2

Отметим, что $\varphi_0(0) = \psi(0)$ и $\varphi_\ell(\ell) = \psi(\ell)$.

Примечание 2. Поскольку задача (5.1) содержит начальные условия по времени, то она является задачей Коши.

Примечание 3. Точно такая же математическая формулировка задачи соответствует распределению температуры в стержне.

5.2 Метод конечных разностей

Нанесем на пространственно-временную область прямоугольную сетку. Пронумеруем узлы сетки:

$i = 0,1, ..., n$ – номер узла по длине (ось x);

$k = 0,1, ...$ – номер узла по времени (ось t).

Таким образом, каждый узел сетки имеет номер, состоящий из двух компонент (i,k). В свою очередь, каждый узел сетки имеет координаты (x_i, t_k).

Обозначим

$u_i^k = u(x_i, t_k)$,

τ – по времени: $\tau = t_{k+1} - t_k$,

h – шаг по длине: $h = x_{i+1} - x_i$,

$f_i^k = f(x_i, t_k)$,

$\varphi_0^k = \varphi_0(t_k), \quad \varphi_\ell^k = \varphi_l(t_k), \quad \psi_i = \psi(x_i)$.

В зависимости от типа разностной схемы для решения задачи теплопроводности различаются **явная схема** и **неявная схема**, каждая из которых имеет свои преимущества и недостатки.

5.2.1. Явная схема

Заменим в дифференциальном уравнении задачи (5.1) для внутренних точек области производные конечно-разностными отношениями. Начальные и краевые условия будем рассматривать в граничных узлах сетки.

$$\begin{cases} \dfrac{u_i^{k+1}-u_i^k}{\tau} = \alpha \dfrac{u_{i-1}^k - 2u_i^k + u_{i+1}^k}{h^2} + f_i^k, & \begin{matrix} i = 1,\ldots,n-1 \\ k = 0,1,\ldots \end{matrix} \\ \left.\begin{matrix} u_0^k = \varphi_0^k \\ u_n^k = \varphi_\ell^k \end{matrix}\right\}, \quad k = 0,1,\ldots - \textit{краевые} \quad \textit{условия} \\ u_i^o = \psi_i, \quad i = 0,1,\ldots,n - \textit{начальные} \quad \textit{условия} \end{cases} \tag{5.2}$$

В результате конечно-разностной аппроксимации получаем в каждой точке сетки одно неизвестное u_i^k и одно условие (граничное или разностное уравнение).

Первую формулу можно переписать в виде

$$u_i^{k+1} = u_i^k + \frac{\tau\alpha}{h^2}(u_{i-1}^k - 2u_i^k + u_{i+1}^k) + \tau f_i^k, i = 1, \ldots, n-1 \tag{5.3}$$

.

Таким образом, по этой формуле имея значения u_i^k для всех i, можно вычислить значения u_i^{k+1} для всех i (без решения уравнений).

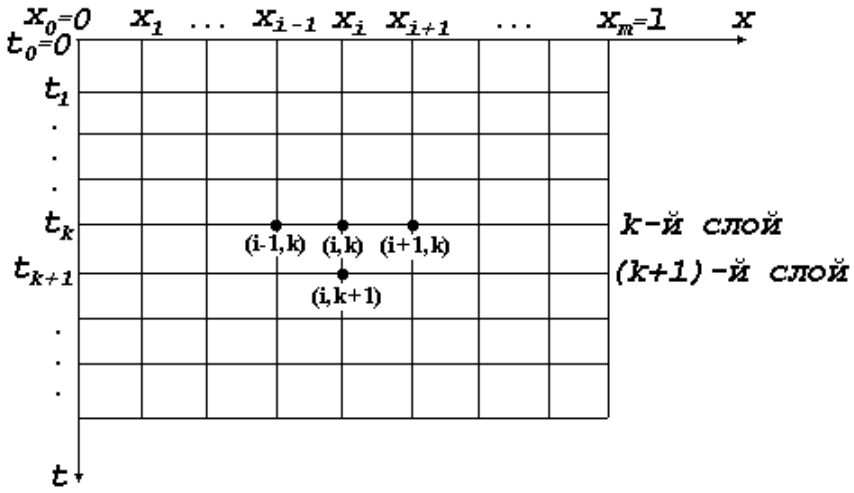

Рис. 5.3 Явная схема

При решении по явной схеме следует учитывать фактор устойчивости счета, который накладывает ограничение на величину шага по времени τ в зависимости от величины шага по x:

$$\tau \le \frac{h^2}{2\alpha} - \textit{условие устойчивости} \qquad (5.4)$$

Нарушение условия устойчивости приводит к неправильному результату. Это ограничение весьма существенно при необходимости решения задачи для большого периода времени.

Примечание. Условие устойчивости, как правило, получается из дополнительных аналитических оценок и носит достаточно общий характер для широкого класса задач. Простой и характерный пример анализа устойчивости, связанной с выбором шага, рассмотрен в теме “Задача Коши” в данном курсе.

5.2.2. Неявная схема

Существует конечно-разностная аппроксимация, счет по которой не требует выполнения условия устойчивости.

$$\begin{cases} \dfrac{u_i^{k+1}-u_i^k}{\tau}=\alpha\dfrac{u_{i-1}^{k+1}-2u_i^{k+1}+u_{i+1}^{k+1}}{h^2}+f_i^{k+1}, & i=1,\ldots,n-1 \\ & k=0,1,\ldots \\ u_0^k=\varphi_0^k \\ u_n^k=\varphi_\ell^k \end{cases}, \quad k=0,1,\ldots - \textit{краевые} \quad \textit{условия} \qquad (5.5)$$

$$u_i^o=\psi_i, \quad i=0,1,\ldots,n - \textit{начальные} \quad \textit{условия}$$

Как видно из записи, аппроксимация второй производной по x осуществляется для точек $(k+1)$-го момента времени.

Первую формулу можно переписать в виде

$$u_i^{k+1}-\frac{\tau\alpha}{h^2}(u_{i-1}^{k+1}-2u_i^{k+1}+u_{i+1}^{k+1})=u_i^k+\tau f_i^{k+1}, \qquad (5.6)$$
$$i=1,\ldots,n-1$$

Следовательно, если все величины u_i^k известны, то величины

$$u_1^{k+1}, u_2^{k+1}, \ldots, u_{n-1}^{k+1}$$

получаем из системы уравнений (5.6) с учетом краевых условий:

$$u_0^{k+1}=\varphi_0^{k+1}$$

и

$$u_n^{k+1}=\varphi_l^{k+1}.$$

Таким образом, начиная с использования начальных условий

$$u_i^o=\psi_i, \quad i=0,1,\ldots,n,$$

последовательно получаем значения неизвестных u_i^k для всех k-ых моментов времени.

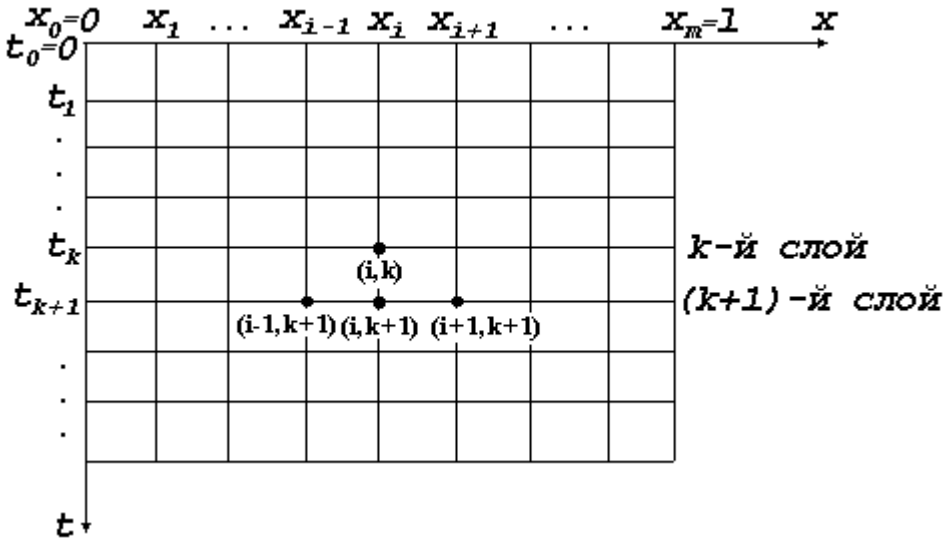

Рис. 5.4 Неявная схема

Примечание. Доказательство устойчивости неявной схемы, как правило, приводится в математических курсах численных методов. Преимущество неявной схемы состоит в том, что она устойчива при любом шаге по времени. Недостатком метода является необходимость решения системы на каждом шаге по времени.

5.2.3. Матричная запись явной схемы

Явная схема (5.2) может быть представлена в виде

$$u_0^{k+1} = \varphi_0^{k+1}, \quad i = 0;$$

$$u_i^{k+1} = u_i^k + \frac{\tau\alpha}{h^2}(u_{i-1}^k - 2u_i^k + u_{i+1}^k) + \tau f_i^k \ , \ i = 1, ..., n-1;$$

$$u_n^{k+1} = \varphi_\ell^{k+1}, \quad i = n.$$

Введем следующие матрицы и векторы

$$E_0 = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & & 0 \end{bmatrix}, \quad A_0 = \begin{bmatrix} 0 & 0 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 0 & 0 \end{bmatrix},$$

$$\bar{u}^k = \begin{bmatrix} u_0^k \\ u_1^k \\ \vdots \\ \vdots \\ u_m^k \end{bmatrix}, \quad \bar{\varphi}^k = \begin{bmatrix} \varphi_0^k \\ 0 \\ \vdots \\ 0 \\ \varphi_\ell^k \end{bmatrix}, \quad \bar{f}^k = \begin{bmatrix} 0 \\ f_1^k \\ \vdots \\ f_{n-1}^k \\ 0 \end{bmatrix}, \quad \bar{\psi} = \begin{bmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \\ \psi_n \end{bmatrix}.$$

Тогда явная схема представима в матричной записи

$$\bar{u}^{k+1} = (E_0 + \frac{\tau\alpha}{h^2} A_0)\bar{u}^k + \bar{\varphi}^{k+1} + \tau\bar{f}^k, \quad k = 0,1,\ldots \quad , \qquad (5.7)$$

при этом $\bar{u}^0 = \bar{\psi}$.

5.2.4. Матричная запись неявной схемы

Неявная схема (5.5) может быть представлена в виде

$u_0^{k+1} = \varphi_0^{k+1}, \quad i = 0;$

$u_i^{k+1} - \frac{\tau\alpha}{h^2}(u_{i-1}^{k+1} - 2u_i^{k+1} + u_{i+1}^{k+1}) = u_i^k + \tau f_i^{k+1} \quad , \quad i = 1, \ldots, n-1;$

$u_n^{k+1} = \varphi_\ell^{k+1}, \quad i = n.$

С учетом принятых обозначений матрично-векторное представление вышеприведенной системы разностных уравнений будет иметь вид

$$(E - \frac{\tau\alpha}{h^2} A_0)\bar{u}^{k+1} = E_0\bar{u}^k + \bar{\varphi}^{k+1} + \tau\,\bar{f}^{k+1} \qquad (5.8)$$

здесь E – единичная матрица $(n+1)$ - порядка

$$E = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

Обозначим

$$A = E - \frac{\tau\alpha}{h^2} A_0$$

Тогда матричная запись решения на каждом шаге по времени по неявной схеме может быть представлена в виде

$$\bar{u}^{k+1} = A^{-1}(E_0 \bar{u}^k + \bar{\varphi}^{k+1} + \tau \bar{f}^{k+1}) \qquad (5.9)$$

при этом $\bar{u}^0 = \bar{\psi}$.

5.3 Решение задачи теплопроводности на MATLAB

Решим задачу (5.1), т.е. найдем распределение температуры **u(x,t)** в стене толщиной **1** и коэффициентом температуропроводности **a=0.001**. На поверхности слева и справа поддерживаются постоянные температуры 110 и 130 соответственно. Начальная температура всюду нулевая.

Применим явную разностную схему, сделав 99 шагов по времени с шагом 5 и разделив стену по толщине на 9 частей. Выведем график распределения температуры для моментов времени 5, 15, 100, 495.

Текст М-файла приведен ниже

```
a=0.001;
n=9;
m=99;
h=1./n; tau=5;
u=zeros(m,n+1);
x=0:h:n*h;
for i=2:n
        u(1,i)=0;
end
t=0:tau:m*tau;
for k=1:m-1
    u(k,1)=110;
    u(k,n+1)=130;
```

```
    for i=2:n
        u(k+1,i)=u(k,i)+tau*a*(u(k,i-1)- ...
            2*u(k,i)+u(k,i+1))/h^2;
    end
end
plot(x,u(1,:),x,u(3,:), x,u(20,:),x,u(89,:));
xlabel('x'); ylabel('y');
grid on
```

График распределения температур приведен на рисунке 5.5

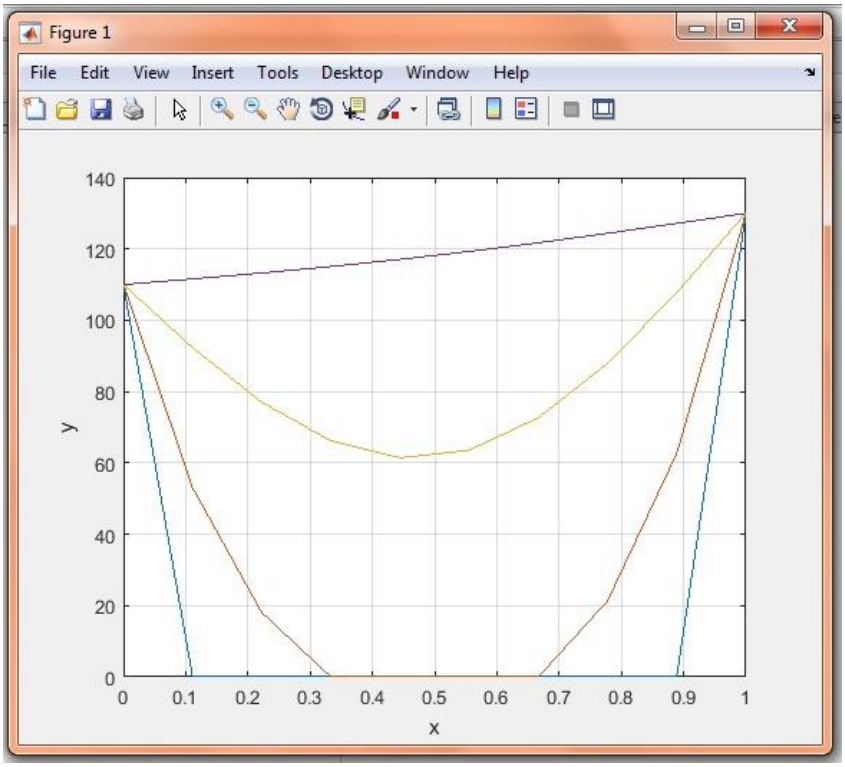

Рис.5.5 Распределение температуры для моментов времени 5, 15,100,495

Лекция 6

ЗАДАЧИ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Введение

Многие практические задачи, связанные с планированием производства, экономические задачи, расчет конструкций и т.д. сводятся к, так называемой, **задаче математического программирования**. Такая задача состоит в вычислении максимума (минимума) **функции цели** $\Phi(x)$, $x=(x_1,x_2,...,x_n)$ при заданных ограничениях $Q_i(x)\le 0$, $i=1,2,...,m$, $m\ge n$. Наиболее простым представителем такой задачи является **задача линейного программирования**.

6.1 Формулировка задачи линейного программирования

Дана линейная целевая функция

$$z(x)=c_1x_1+c_2x_2+...+c_nx_n \qquad (6.1)$$

и система m линейных неравенств-ограничений ($m\ge n$)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \le b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \le b_2 \\ - - - - - - - - - - - - - - - - - - \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \le b_m \end{cases}, \qquad (6.2)$$

а также дополнительные ограничения

$$x_1 \ge 0,\ x_2 \ge 0,\ \ldots,\ x_n \ge 0. \qquad (6.3)$$

Требуется найти максимум функции $z(x)$ при выполнении заданных ограничений (6.2)-(6.3).

Можно записать (6.1) - (6.3) в стандартной и матрично-векторной форме.

Короткая (стандартная) форма

$$z(x) = \sum_{i=1}^{n} c_i x_i \Rightarrow \max \qquad (6.1')$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i,\ i = 1,\ 2,\ \ldots,\ m \qquad (6.2')$$

$$x_j \ge 0,\ j = 1,\ 2,\ \ldots,\ n \qquad (6.3')$$

Матрично-векторная форма

$$z(c, x) \Rightarrow \max \qquad (6.1'')$$

$$Ax \le b \qquad (6.2'')$$

$$x \ge 0 \qquad (6.3'')$$

где

c – вектор целевой функции,

b – вектор ограничений,

A – матрица ограничений.

6.2. Пример формулировки и решения задачи линейного программирования

Задачи линейного программирования достаточно содержательны с практической точки зрения и при этом, как правило, имеют решение в классе точных методов, представителем которых является, например, *симплекс-метод*.

Рассмотрим практический пример из строительства.

Имеется растворный узел, производящий бетон двух видов (в зависимости от расхода цемента, песка и щебня). Исходные данные по работе такого узла могут быть представлены таблицей

Вид сырья	Расход сырья на единицу продукции		Запас сырья
	1й тип бетона	2й тип бетона	
Цемент	0.25	0.25	1.5
Песок	0.25	0.5	2.5
Щебень	0.5	0.25	2.5
Цена ед. продукции	2	1	

Пусть

x_1 – количество выпущенного бетона 1-го типа,

x_2 – количество выпущенного бетона 2-го типа,

тогда

$z = 2x_1 + x_2$ – доход растворного узла.

При этом имеет место следующий расход материала:

$$\begin{aligned} &\text{цемента:} \quad 0.25x_1 + 0.25x_2 \le 1.5, \\ &\text{песка:} \quad\;\; 0.25x_1 + 0.5x_2 \le 2.5, \\ &\text{щебня:} \quad\;\; 0.5x_1 + 0.25x_2 \le 2.5. \end{aligned} \qquad (*)$$

В правой части неравенств (*) присутствует ограничение, связанное с реальным запасом материалов в растворном узле.

Задача состоит в таком планировании производства (то есть определении x_1 и x_2 – плана выпуска), при котором величина дохода (z) максимальна, т.е. найти точку максимума функции z

$$z = 2x_1 + x_2$$

при ограничениях

$$\begin{cases} x_1 + x_2 \le 6, \\ x_1 + 2x_2 \le 10, \\ 2x_1 + x_2 \le 10, \\ x_1 \ge 0, \\ x_2 \ge 0. \end{cases} \qquad (**)$$

Здесь неравенства (*) для упрощения расчета умножены на число «4». Последние два неравенства в (**) являются очевидными, поскольку в нашем случае объем выпускаемой продукции не может быть отрицательным.

В **матричной форме**: найти вектор x, при котором функция $z = (c, x)$ достигает максимума, при ограничениях $Ax \le b$ и $x \ge 0$

где

$$c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad b = \begin{pmatrix} 6 \\ 10 \\ 10 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

6.2.1. Геометрическая интерпретация

Как следует из аналитической геометрии, каждое из неравенств-ограничений (**) задает некоторую полуплоскость. Их совокупность определяет некоторый выпуклый многоугольник Ω. Он представлен на рис.6.1. Его называют **многоугольником ограничений**. Очевидно, что искомые значения x_1 и x_2 принадлежат этому многоугольнику.

Среди точек области Ω необходимо отыскать такую, в которой функция z принимала бы максимальное значение.

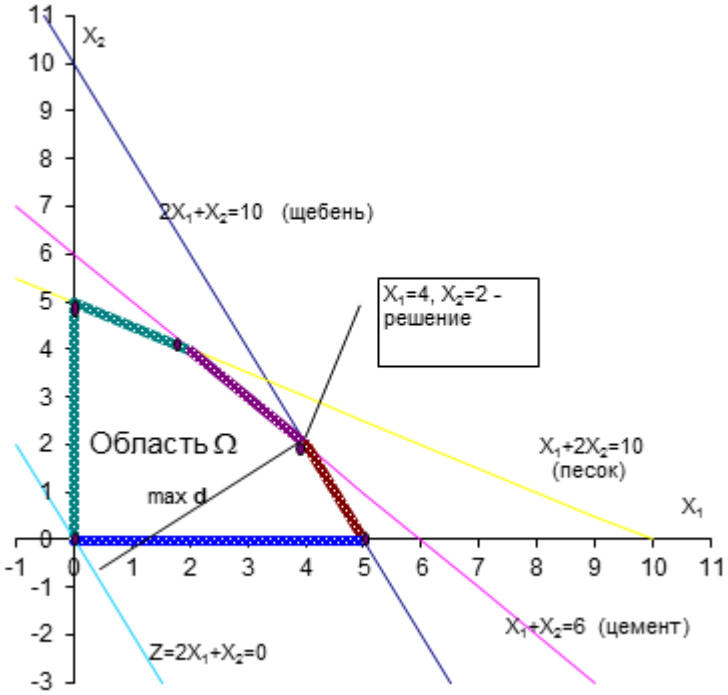

Рис. 6.1.

Выясним геометрический смысл целевой функции z.

Из аналитической геометрии известно, что расстояние d от точки (x_1, x_2) до прямой $z = 0$, где $z = ax_1 + bx_2 + c$, определяется по формуле

$$d = \frac{|ax_1 + bx_2 + c|}{\sqrt{a^2 + b^2}} = \frac{1}{\sqrt{a^2 + b^2}} |z| .$$

Поскольку в рассматриваемой задаче $z \geq 0$, то z пропорциональна d и максимум значения z достигается в точке, максимально удаленной от прямой $z = 0$.

Теперь задачу можно поставить так.

Среди точек многоугольника Ω найти точку, наиболее удаленную от прямой

$$z = 2x_1 + x_2 = 0 .$$

Из рис. 6.1 видно, что такой точкой будет точка пересечения прямых:

$$x_1 + x_2 = 6,$$
$$2x_1 + x_2 = 10,$$

т.е. одна из вершин многоугольника. Решая эту систему уравнений, находим координаты искомой точки:

$$x_1 = 4\ , \quad x_2 = 2\ .$$

Следовательно, в соответствии оптимальному плану (наибольший доход) нужно выпускать 4 единицы бетона 1-го типа и 2-го единицы бетона 2 типа. При этом будет получен доход

$$z = 2 \cdot 4 + 2 = 10$$

Предложенный метод решения – **геометрический**. Он пригоден для случая лишь двух-трех переменных.

Перечислим некоторые широко употребляемые термины задач линейного программирования:

Многогранник ограничений – область допустимых значений.

Допустимое решение – точка, удовлетворяющая всем ограничениям.

Опорное решение – одна из вершин многогранника ограничений. Оно удовлетворяет всем ограничениям и, по крайней мере, n из них обращает в равенство.

Оптимальное решение – точка, удовлетворяющая всем ограничениям, в которой функция цели принимает максимальное значение.

Замечание 1. Из теории задач линейного программирования известно, что оптимальное решение достигается в одной из вершин многогранника ограничений. Однако прямой перебор вершин занимает слишком много времени. Поэтому разработаны методы, которые позволяют вести направленный перебор вершин (с увеличением величины z). Из таких методов наиболее используемым является **симплекс-метод**, алгоритм которого излагается в специальных курсах.

Замечание 2. Многогранник ограничений может быть неограниченным или сами ограничения могут быть противоречивыми. Эти случаи требуют специального рассмотрения.

Замечание 3. В реальной задаче линейного программирования помимо ограничений типа неравенств могут быть ограничения в виде строгих равенств. Такое равенство можно представить как совокупность двух неравенств. Например, равенство

$$a_{kl}\, x_l + \ldots + a_{kn}\, x_n = b_k$$

может быть записано в виде двух неравенств

$$a_{kl}\, x_l + \ldots + a_{kn}\, x_n \geq b_k\,,$$

$$a_{kl}\, x_l + \ldots + a_{kn}\, x_n \leq b_k\ .$$

6.3 Средства MATLAB для решения задач линейного программирования

Для решения задач линейного программирования в системе MATLAB используется функция **linprog**, соответствующая область поиска для которой задается следующими условиями:

A*x<=b — линейные неравенства (A-матрица, b- вектор);

Aeq*x=beq - линейные уравнения (Aeq - матрица, beq - вектор);

lb<=x<=ub - ограничения на координаты (lb, ub - векторы);

Целевая функция c'*x в **linprog** задается вектором коэффициентов c.

Перечислим ниже формы обращения к функции **linprog**

x= linprog(f,A,b,Aeq,Beq)

x= linprog(f,A,b,Aeq,Beq,lb,ub)

x= linprog(f,A,b,Aeq,Beq,lb,ub,x0)

x= linprog(f,A,b,Aeq,Beq,lb,ub,x0,options)

[x,fval]= linprog(. . .)

[x,fval,exitflag]= linprog(. . .)

[x,fval,exitflag,output]= linprog(. . .)

[x,fval,exitflag,output,lambda]= linprog(. . .)

где

x — искомый вектор;

c — вектор коэффициентов функции цели;

fval — искомое минимальное значение функции цели;

exitflag — параметр, характеризующий вычислительный процесс,(если его значение больше нуля, то результат найден с требуемой точностью, нуль - достигнуто максимальное значение итераций, меньше нуля - решение не найдено);

output — структура, содержащая сведения о процессе оптимизации;

lambda — структура, содержащая сведения о множителях Лагранжа для решения x.

x0— стартовая (начальная) точка.

Отсутствующие ограничения в списке параметров задаются квадратными скобками.

В функции **linprog** по умолчанию используется так называемый прямо-двойственный алгоритм, при использовании которого одновременно решается как исходная задача линейного программирования, так и двойственная к ней (при этом решение двойственной задачи не выдается).

Решим задачу линейного программирования (**) с использованием функции **linprog**

Текст М-файла:

```
clear
%Задание количества неизвестных
n=input('Введите n=');
%Задание количества ограничений
m= input('Введите m=');
```

```
%Формирование вектора коэффициентов функции цели
c=[2;1];
%Формирование матрицы  коэффициентов ограничений
A=[1  1; 1  2; 2  1];
%Формирование вектора правых частей ограничений
b=[6; 10 ; 10];
%Распечатка исходных данных
disp('Вектор коэффициентов функции цели, c');
fprintf('%12.4f \n',c);
disp('Матрица  коэффициентов ограничений, A');
for k=1:m
    fprintf('%12.4f',A(k,:));
    fprintf('\n')
end
disp('Вектор правых частей ограничений, b ');
fprintf('%12.4f \n',b);

%Задание вектора ограничений снизу на координаты
lb= zeros(2,1);
%Обращение к функции MATLAB для решения зпадачи
%линейного программирования
[x,z]=linprog(-c,A,b,[],[],lb);

%Распечатка результатов
disp('Вектор решения, x');
fprintf('%12.4f \n',x);
fprintf('Максимальное значение функции цели,z=%12.4f',
z);
```

Результаты расчета

```
Введите n= 2
Введите m= 3
Вектор коэффициентов функции цели, c
      2.0000
      1.0000
Матрица  коэффициентов ограничений, A
      1.0000      1.0000
      1.0000      2.0000
      2.0000      1.0000
Вектор правых частей ограничений, b
      6.0000
     10.0000
     10.0000
Optimization terminated.
Вектор решения, x
```

4.0000
2.0000

Лекция 7

МЕТОД КОНЕЧНЫХ ЭЛЕМЕНТОВ (МКЭ)

(На примере краевой задачи для обыкновенного дифференциального уравнения изгиба растянуто-изогнутой балки)

Введение

Метод конечных элементов в настоящее время является наиболее распространенным методом расчета сложных конструкций. На его основе разработаны мощные и универсальные пакеты программ, как в России, так и за рубежом. В рамках данной темы частично излагается основная идея метода.

7.1. Соответствие задачи определения минимума функционала и решения системы уравнений.

Пусть задан функционал

$$\Phi(x) = \frac{1}{2}(Ax, x) - (b, x) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j - \sum_{i=1}^{n} b_i x_i \qquad (7.1)$$

где

A – симметричная и положительно определенная матрица: $A = A^{\mathrm{T}}$ и $(Ax, x) > 0$ при всех x,

b – заданный вектор.

Требуется найти $\min \Phi(x)$ на множестве векторов x.

Условие минимума функционала (минимума функции нескольких переменных) имеет вид:

$$\begin{cases} \dfrac{\partial \Phi(x)}{\partial x_i} = \sum\limits_{j=1}^{n} a_{ij} x_j - b_i = 0 \\ i = 1,2,\ldots,n \end{cases} \quad \text{или} \quad Ax = b \qquad (7.2)$$

Решение (7.2) является точкой минимума функционала (7.1), то есть существует взаимно однозначное соответствие между задачей о минимуме функционала и решением системы линейных уравнений с симметричной матрицей.

7.2. Вариационная постановка задачи об изгибе растянуто-изогнутой балки

Из раздела математики «Вариационное исчисление» следует, что задача об изгибе растянуто-изогнутой балки может быть представлена задачей на минимум следующего функционала:

$$\Phi(y(x)) = \frac{1}{2}\int_0^{\ell} (EJy'(x)^2 + Py(x)^2)dx - \int_0^{\ell} M(x)y(x)dx \qquad (7.3)$$

где

$y(x)$ – функция прогиба балки,

$EJ = EJ(x)$ – жесткость балки,

$q(x)$ – заданная нагрузка на балку,

P – заданная сила,

$M(x)$ – изгибающий момент в балке (известен, поскольку балка статически определимая),

ℓ – длина балки.

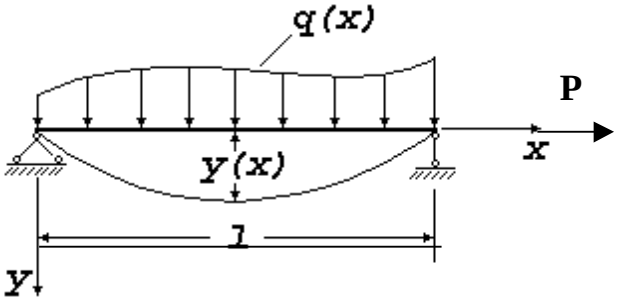

Рис. 7.1

Задача состоит в определении функции $y(x)$, для которой функционал $\Phi(y)$ принимает минимальное значение ($\min \Phi(y)$). При этом функция $y(x)$ должна удовлетворять дополнительному условию

$y(0) = y(\ell) = 0$ – **шарнирное опирание**.

Из курса «Вариационное исчисление» следует, что такая задача на минимум эквивалентна следующей краевой задаче:

$$\begin{cases} -EJy'' + py = M(x), \quad x \in (0, \ell), \\ y(0) = y(\ell) = 0 \qquad - \textit{ краевые условия.} \end{cases} \tag{7.3а}$$

Большинству технических задач, как правило, также соответствуют две эквивалентные постановки: вариационная (задача на минимум функционала) и краевая (представленная дифференциальным уравнением и краевыми условиями), имеющих одно и то же решение. По целому ряду соображений вариационная постановка предпочтительнее, поскольку она приводит к более простым и универсальным алгоритмам решения.

7.3 Решение вариационной задачи

Разобьем отрезок $(0,\ell)$ на $(N-1)$ часть (конечных элементов), в соответствии с рис 7.2. Введем обозначения

x_i – координата начала i-го конечного элемента (точка разбиения),

$h_i = x_{i+1} - x_i$ – длина i-го конечного элемента,

$y_i = y(x_i)$ – значение искомой функции в i-ой точке разбиения.

Представим $\Phi(y)$ в виде суммы

$$\Phi(y) = \sum_{i=1}^{N-1} \Phi_i(y) \qquad (7.4)$$

$$\Phi_i(y(x)) = \frac{1}{2}\int_{x_i}^{x_{i+1}} (EJy'(x)^2 + py(x)^2)dx - \int_{x_i}^{x_{i+1}} M(x)y(x)dx$$

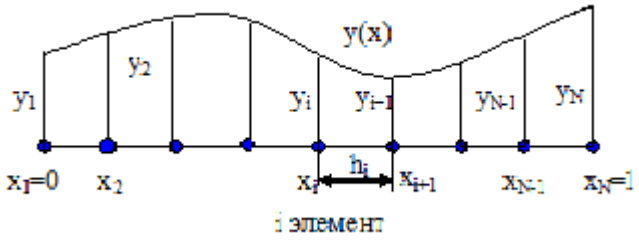

Рис. 7.2.

Примем, что внутри каждого i-го элемента ($x \in (x_i, x_{i+1})$), функция $y(x)$ линейная и принимает на краях элемента соответственно значения y_i и y_{i+1} (рис. 7.3):

$y(x) = y_i(x) = (1-z)y_i + zy_{i+1}$,

где $z = \dfrac{x - x_i}{h_i}$, $x \in (x_i, x_{i+1})$

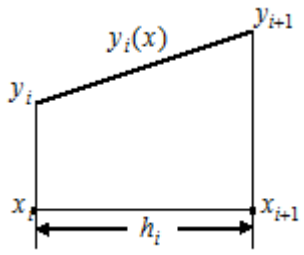

Рис. 7.3

Обозначим

$\psi_1 = 1 - z$, $\psi_2 = z$,

$\upsilon_1^i = y_i$, $\upsilon_2^i = y_{i+1}$.

Тогда

$$y_i(x) = \psi_1 \upsilon_1^i + \psi_2 \upsilon_2^i = (\overline{\psi}, \overline{\upsilon}^i) \qquad (7.5)$$

где

$\psi_1 = 1 - z$ и $\psi_2 = z$ – называются **функциями формы** (рис. 7.4),

при этом производные функций форм имеют вид: $\psi_1' = -1$, $\psi_2' = 1$;

$\upsilon_1^i = y_i$ и $\upsilon_2^i = y_{i+1}$ – называются **локальными неизвестными i-го элемента;**

$\overline{\psi} = \begin{pmatrix} \psi_1 \\ \psi_2 \end{pmatrix}$ и $\overline{\upsilon}^i = \begin{pmatrix} \upsilon_1^i \\ \upsilon_2^i \end{pmatrix}$ – векторы функций форм и локальных неизвестных, соответственно.

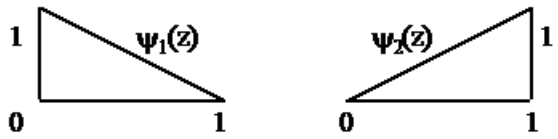

Рис.7.4

Подставляя $y_i(x)$ в $\Phi_i(y)$ и делая замену переменных, получим

$$\Phi_i(y) = \frac{1}{2}\int_0^1 \left(\frac{EJ}{h_i}(\psi_1'\upsilon_1^i + \psi_2'\upsilon_2^i)^2 + h_i\, p(\psi_1\upsilon_1^i + \psi_2\upsilon_2^i)^2 \right) dz -$$

$$- h_i \int_0^1 M(x)(\psi_1\upsilon_1^i + \psi_2\upsilon_2^i)dz$$

Вычисляя интегралы и приводя подобные члены, получим:

$$\Phi_i(\bar{y}) = \Phi_i(\bar{\upsilon}^i) = \frac{1}{2}\sum_{s=1}^{2}\sum_{t=1}^{2} K_{st}{}^i \upsilon_s^i \upsilon_t^i - \sum_{s=1}^{2} R_s \upsilon_s^i = \frac{1}{2}(K^i \bar{\upsilon}^i, \bar{\upsilon}^i) - (\bar{R}^i, \bar{\upsilon}^i) \qquad (7.6)$$

где матрица K^i называется **матрицей жесткости i-го конечного элемента**. В свою очередь ее элементы вычисляются по формуле

$$K_{st}^i = \int_0^1 (\frac{EJ}{h_i}\psi'_s\,\psi'_t + h_i\, p\psi_s\psi_t)dz, \quad s = 1,2, \quad t = 1,2.$$

Вектор $\bar{R}^i$ называется **вектором нагрузки i-го элемента**. Его компоненты вычисляются по формуле

$$R_s^i = h_i \int_0^1 M(z)\psi_s(z)dz, \quad s = 1,2.$$

Опуская промежуточные выкладки, получим

$$K^i = \frac{EJ}{h_i}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \frac{ph_i}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \qquad (7.7)$$

$$\bar{R}^i = \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \frac{M_i h_i}{2}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad (7.8)$$

где $M_i = M\left(x_i + \frac{h_i}{2}\right)$

7.3.1. Глобальные матрица жесткости и вектор нагрузки

Общий функционал, представляющий сумму функционалов по элементам примет вид:

$$\Phi_h(\bar{y}) = \frac{1}{2}\sum_{i=1}^{N-1}(K^i\bar{\upsilon}^i, \bar{\upsilon}^i) - \sum_{i=1}^{N-1}(\bar{R}^i, \bar{\upsilon}^i) \qquad (7.9)$$

где $\bar{y} = (y_1, y_2, ..., y_N)^{\mathrm{T}}$.

Из определения вектора $\bar{\upsilon}^i$ следует, что $\upsilon_s^i = y_{i+s-1}$. Заменяя в (7.9) компоненты векторов $\bar{\upsilon}^i$ на компоненты вектора $\bar{y}$ и опуская промежуточные выкладки, получим:

$$\Phi_h(\bar{y}) = \frac{1}{2}(K\,\bar{y}, \bar{y}) - (\bar{R}, \bar{y}),$$

где

K называется **глобальной матрицей жесткости**.

Элементы матрицы K вычисляются по формулам:

$$K_{ij} = \begin{cases} K_{21}^{i-1} & при \quad j = i-1 \\ K_{22}^{i-1} + K_{11}^{i} & при \quad j = i \\ K_{12}^{i} & при \quad j = i+1 \\ 0 & при \quad |i-j| > 1 \end{cases} \qquad (7.10)$$

Общая структура:

$$K = \begin{bmatrix} K_{11}^1 & K_{12}^1 & & & & & \\ K_{21}^1 & K_{22}^1 + K_{11}^2 & K_{12}^2 & & & & \\ & K_{21}^2 & K_{22}^2 + K_{11}^3 & K_{12}^3 & & & \\ & & K_{21}^3 & \ddots & \ddots & & \\ & & & & K_{21}^{N-2} & K_{22}^{N-2} + K_{11}^{N-1} & K_{12}^{N-1} \\ & & & & & K_{21}^{N-1} & K_{22}^{N-1} \end{bmatrix}.$$

R называется **глобальным вектором нагрузки**.

Компоненты вектора R вычисляются по формулам:

$$R_i = R_2^{i-1} + R_1^i \tag{7.11}$$

Общая структура:

$$\overline{R} = \begin{bmatrix} R_1^1 \\ R_2^1 + R_1^2 \\ R_2^2 + R_1^3 \\ \vdots \\ \vdots \\ R_2^{N-2} + R_1^{N-1} \\ R_2^{N-1} \end{bmatrix}.$$

При программировании вычисления элементов глобальной матрицы жесткости и правой части наиболее удобными являются формулы (формулы «конечных вкладов»):

$$K_{i+s-1,\,i+t-1} = \sum_{i=1}^{N-1} K_{st}^i$$

$$R_{i+s-1} = \sum_{i=1}^{N-1} R_s^i \tag{7.11}$$

$$s = 1,2\,, \quad t = 1,2\,, \quad i = 1,2,\ldots,N$$

Замечание. На практике, как правило, локальные матрицы жесткости и векторы нагрузки (7.7)-(7.8) для конкретного вида конструкции известны заранее (существует специальная библиотека конечных элементов). Они соответствуют математическому выражению под знаком интеграла в исходном функционале и конфигурации конечных элементов. Поэтому основная задача расчетчика состоит в разбиении конструкции на конечные элементы и

формировании глобальной матрицы жесткости и вектора нагрузки (хотя и здесь имеются стандартные алгоритмы).

Решение задачи получаем из решения глобальной системы

$$K\bar{y} = \bar{R} \qquad (7.12)$$

7.3.2. Учет закреплений

Для того чтобы удовлетворить условию $y_1 = y_N = 0$ следует приравнять нулю элементы первых и последних строк и столбцов матрицы K, а затем положить $K_{11} = K_{NN} = 1$ и $R_1 = R_N = 0$.

7.4 Решение задач методом конечных элементов в MATLAB

Решим задачу (7.3) об изгибе растянуто-изогнутой балки.

Рассмотрим следующие исходные данные:

EJ=3/8; P=576; l=1, M(x)=0.6(2EJ+Px(l-x))

```
function mkel
%Задание количества неизвестных
n=input('Введите n=');
%Задание длины балки
dl=input('введите длину балки dl=');
EJ=3/8;
%Задание осевой силы
P=576;
% Формирование глобальной матрицы жесткости
%и вектора нагрузок
Kg=zeros(n,n);
Rg=zeros(n,1);
h=dl/(n-1);
Ph=P*h/6;
EJh=EJ/h;
K0=EJh+2*Ph;
K1=Ph-EJh;
Kg(1,1)=1.; Kg(n,n)=1.;
for i=2:n-1
    x=(i-1.5)*h;
```

```
    Rg(i)=(M(x,g,s,dl,EJ,P)+M(x+h,g,s,dl,EJ,P))*h/2;
    if(i>2)
        Kg(i,i-1)=K1;
    end
    Kg(i,i)=2*K0;
    if (i<n-1)
        Kg(i,i+1)=K1;
    end
end
%Распечатка глобальной матрицы жесткости
%и вектора нагрузок

disp('Глобальная матрица жесткости Kg');
for i=1:n
    fprintf('%6.2f ',Kg(i,:));
    fprintf('\n');
end
disp('Глобальный вектор нагрузок Rg');
fprintf('%12.4f \n',Rg);

%Решение разрешающей системы линейных алгебраических
% уравнений, распечатка результата
y=Kg\Rg;
disp('Вектор решения y');
fprintf('%12.4f \n',y);


function Res=M(x,g,s,dl,EJ,P)
Res=(g+s)/25.*(2*EJ+P*x*(dl-x));
end
end
```

```
Введите n=9
введите длину балки dl=1

Глобальная матрица жесткости Kg
1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
0.00  54.00   9.00   0.00   0.00   0.00   0.00   0.00   0.00
0.00   9.00  54.00   9.00   0.00   0.00   0.00   0.00   0.00
0.00   0.00   9.00  54.00   9.00   0.00   0.00   0.00   0.00
0.00   0.00   0.00   9.00  54.00   9.00   0.00   0.00   0.00
0.00   0.00   0.00   0.00   9.00  54.00   9.00   0.00   0.00
0.00   0.00   0.00   0.00   0.00   9.00  54.00   9.00   0.00
0.00   0.00   0.00   0.00   0.00   0.00   9.00  54.00   0.00
0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   1.00
```

Глобальный вектор нагрузок Rg

```
      0.0000
      4.6125
      7.9875
     10.0125
     10.6875
     10.0125
      7.9875
      4.6125
      0.0000
```

Вектор решения y

```
      0.0000
      0.0665
      0.1133
      0.1414
      0.1508
      0.1414
      0.1133
      0.0665
      0.0000
```

Лекция 8
ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ СТРОИТЕЛЬНЫХ ОБЪЕКТОВ

8.1. Конструкция и ее расчетная схема

8.1.1. Понятие о расчетной схеме конструкции.

Расчетный анализ конструкции начинается с постановки практической задачи. Вначале важно (и в действительности не просто) установить, на какой вопрос мы хотим ответить в результате анализа конструкции. Затем следует определить, какие факторы в рассматриваемой проблеме значимо (существенно) влияют на изучаемое проявление конструкции, а какими из факторов, напротив, можно пренебречь. Такого рода упрощение задачи производится всегда – расчет реальной конструкции и учет ее фактических свойств возможны лишь с определенной степенью приближения.

Реальную конструкцию, «освобожденную» от всех несущественных особенностей и представленную таким образом в некоторой идеализированной (упрощенной) форме называют её *физической моделью*. Некоторые методы подобной идеализации (схематизации) получили широкое рапространение на практике и имеют достаточно общий характер, например, идеализация материала в виде сплошной среды; предположение (гипотеза) об однородности материала и др. Далее, приведение геометрической формы рассматриваемого объекта, его элементов и их взаимосвязей к стандартным упрощенным схематическим решениям как, например, стержни, пластины или оболочки; жёсткие, упругие и шарнирные соединения, а также схематизация внешних силовых воздействий и др. приводят к созданию расчётного аналога конструкции – к ее *расчетной схеме*. Разумеется, существуют и другие, вполне конкретные, связанные с каждой рассматриваемой задачей, методы схематизации. В целом, следует подчеркнуть, что во всех случаях выбор

расчетной схемы является исключительно важным элементом анализа, и, по существу, одним из наиболее характерных черт инженерного искусства (научиться которому можно лишь в процессе практической работы), характеризующей уровень профессионального мастерства расчетчика [119]. Следует отметить, что после формулировки расчетной схемы (возможно даже в достаточно общих чертах) наступает этап ее детального описания в форме, пригодной для выполнения расчетного анализа.

8.1.2. Некоторые аспекты формирования расчетной схемы.

Уже на самой начальной стадии создания расчетной схемы необходимо определиться, будет ли выполняться т.н. *линейный расчет* конструкции (подразумевающий только линейные зависимости всех участвующих в расчете параметров этой конструкции) или, напротив расчет будет *нелинейный*, следует ли учитывать силы инерции и выполнять *динамический расчет* или же можно ограничиться *статическим* анализом.

Об ожидаемом поведении конструкции традиционно судят на основании накопленного опыта и инженерной интуиции, в связи с чем принимаемые решения подлежат апостериорной оценке. Так, например, если во всех математических выражениях, описывающих поведение рассматриваемой системы, можно пренебречь производными по времени, то формулируется *стационарная задача*, и, следовательно, выполняется анализ поведения соответствующей неподвижной системы. В задачах динамики сооружений, когда существенную роль играют силы инерции, пропорциональные ускорениям масс, а также при учёте вязких свойств материалов, когда учитываются их скорости, следует анализировать движущуюся систему.

Нелинейные задачи могут быть связаны с эффектами, возникающими при изменении геометрии системы под нагрузкой (*геометрическая нелинейность*), отсутсвием линейных пропорций между напряжениями и деформациями (*физическая нелинейность*), возможным включением в работу и выключением

из работы соединений при действии нагрузки на систему, например, односторонних связей (*конструктивная нелинейность*) или с эффектами, определяемыми переменностью структуры системы в процессе ее создания, эксплуатации и демонтажа (*генетическая нелинейность*).

Все вышеперечисленные особенности ожидаемого поведения конструкции, очевидно, сказываются на выборе расчетной схемы.

8.1.3. О формировании конечно-элементной расчетной модели конструкции.

Весьма серьезным и непростым вопросом является разбиение системы на конечные элементы, т.е., иными словами, на стандартные части, из которых (и только из которых) должна состоять вся *т.н. расчетная модель* системы. Поясним, что идеализация конструкции должна быть выполнена в форме, адаптированной для использования метода конечных элементов, а именно: система должна быть представлена в виде набора тел стандартного типа (стержней, пластин, оболочек и др.), уже называемых конечными элементами (эти конечные элементы должны быть заранее изучены) и присоединенных в узловых точках. Узел в расчетной схеме метода конечных элементов традиционно представляется в виде абсолютно жесткого тела исчезающе малых размеров (конечные элементы считаются присоединенными только к узлам расчетной схемы).

Избыточно мелкое дробление приводит к увеличению времени расчета и сопряжено с запросами на использование значительных ресурсов памяти компьютера для хранения и обработки данных. Отметим, что в подобных ситуациях могут проявляться эффекты неустойчивости самого вычислительного процесса. Слишком же грубое дробление на конечные элементы может привести к потере точности результатов, особенно в тех случаях, когда моделируется поведение пластинчатых или оболочечных конструкций.

Каких-либо общих, универсальных рекомендаций по выбору оптимального уровня дробления системы на конечные элементы не существует. Существующие «оценки сходимости» имеют асимптотический характер и зачастую являются слишком абстрактными для конструктивного использования в практических приложениях для конкретных расчетных случаев. В этой связи приходится полагаться, прежде всего, на опыт и результаты некоторых контрольных расчетов, выполняемых для одной и той же конструкции при различных схемах разбиения на конечные элементы (т.е. при различных вариантах конечноэлементных аппроксимаций). Кроме того, могут быть рекомендованы приемы, связанные с проведением последовательной серии расчетов некоторых фрагментов системы с введением на этих фрагментах более детального разбиения на конечные элементы.

8.1.4. Нагрузки и воздействия.

Взаимодействие системы с окружающей средой традиционно представляется в расчетных моделях в виде *нагрузок* или *воздействий*, приложенных к узлам системы (*узловых нагрузок*) или к внутренним точкам конечных элементов (*местных нагрузок*). Местные нагрузки могут быть силами и моментами, сосредоточенными или распределенными по линиям, площадям и объемам. Иногда загружение системы представляется в виде температурных воздействий на элементы или в виде заданных смещений в узлах (т.е. эти воздействия проявляются не как силовые, а как кинематические факторы).

Если воздействия меняются во времени, то вызванные ими ускорения масс системы приводят к появлению *инерционных сил*. В случаях, когда силами инерции пренебречь нельзя, принято говорить о *динамическом характере воздействия*, однако не следует забывать, что отнесение воздействий к виду статических или динамических связано не только с их собственными свойствами, но и с *инерционными характеристиками системы*.

В практике проектирования используются *нормативные и расчетные значения нагрузок* (используемые в расчетах разного рода), причем переход от одних к другим выполняется с помощью системы соответствующих коэффициентов. Совокупность нагрузок и воздействий, одновременно приложенных к системе и рассматриваемых совместно, называется ее *загружением*. Заметим, что иногда к общему загружению относят лишь ту часть одновременно приложенных нагрузок, которая связана с общим происхождением или же имеет какие-то другие общие свойства. Если впоследствии возникнет необходимость учета эффекта совместного действия нескольких загружений такого рода (их сумму, взятую с некоторыми коэффициентами), то говорят о *комбинации загружений* и, соответственно, о коэффициентах такой комбинации.

В силу того, что загружения могут в различные моменты времени образовывать различные комбинации, и возможное количество таких комбинаций (сочетаний) достаточно велико, возникает достаточно нетривиальная задача отыскания таких сочетаний внешних воздействий, которые приводят к наиболее неблагоприятным послествиям для некоторого проверяемого элемента, его сечения или конструкции в целом. В описанном случае имеется в виду отыскание так называемого *расчетного сочетания усилий* (*РСУ*). Решая данную задачу, следует помнить о естественной логической связи между загружениями, определяемыми природой действующих на систему нагрузок или же указанимями нормативных документов. Такие логические связи иногда определяют невозможность одновременного действия (несовременстность) некоторых нагрузок. В других случаях, напротив, требуется обязательный учет какого-либо загружения при рассмотрении вполне определенного другого загружения, хотя связь такого рода может и отсутствовать.

Необходимо отметить, что использование комбинаций загружений или же отыскание РСУ основано на принципе суперпозиции и, следовательно, справедливо лишь для линейных систем.

8.2. Понятие о математическом и компьютерном моделировании конструкций, зданий и сооружений

8.2.1. Актуальность математического и компьютерного моделирования в строительстве.

Мировой и отечественный опыт, отраженный в многочисленных трудах конференций, публикациях и монографиях, известные события последних лет в крупных городах мира свидетельствуют: проблема обеспечения безопасности строительных объектов является актуальной, наукоемкой и, к сожалению, сегодня далекой от практического решения. Эффективное, экономически оправданное решение соответствующих задач в развитых странах осуществляется на основе прогнозного математического и комьютерного моделирования состояний (газо- и гидродинамического, теплового, статического и динамического напряженно-деформированного и др.) ответственных объектов инфраструктуры и, при необходимости, их конгломератов с использованием развитых программно-алгоритмических комплексов, реализующих численные методы механики (гидро- и газодинамики, механики твердого тела и др.) Современная концепция требует, чтобы математические модели сопровождали объекты на всех этапах их зарождения (проектирования и строительства) и жизни (эксплуатации, ремонта и реконструкции), обеспечивая адекватный и полный анализ и прогноз состояния в составе компьютерных информационно-диагностических систем мониторинга.

Последние годы указанный подход находит понимание и в Российской Федерации, где проектируются, строятся и эксплуатируются большепролетные

сооружения, высотные многофункциональные здания и другие ответственные строительные объекты, уникальные и по общемировым меркам. В силу пионерного характера они не имеют прямых аналогов и пока еще слабо обеспечены действующими нормативными требованиями и методиками. С другой стороны, все нарастающее значение имеет проблема износа основных фондов, в частности, старения сооружений и коммуникаций. До последнего времени, невыгодно отличаясь от сложившейся системы обязательного, четко нормативно прописанного обоснования опасных промышленных объектов (таких как, атомной и гидроэнергетики), глубина математического и компьютерного моделирования состояния социально ответственных строительных объектов была недостаточной и зависела от множества весьма субъективных факторов (среди которых, практика нежелания инвестора идти на соответствующие затраты, долгожданная «свобода» воплощения самых смелых в инженерном смысле архитектурных замыслов). Отметим, что подобная практика привела как к известным трагическим последствиям-катастрофам (например, обрушение панельного здания на Мичуринском проспекте в городе Москве, купола спортивно-оздоровительного комплекса «Трансвааль-парк» в городе Москве), так и к прослеживаемой тенденции ухудшения экологического состояния (распространение вредных газов, повышенная вибрация от транспорта, ветровых нагрузок и пр.).

8.2.2. Классификация объектов и задач, требующих математического и компьютерного моделирования.

Таким образом, нормативные требования, методическое и программно-алгоритмическое обеспечение для обоснования комплексной безопасности строительных объектов находятся в стадии формирования. Не претендуя на строгость и полноту классификации, можно выделить некоторые группы объектов и задач, требующих построения адекватных прогнозных

математических и компьютерных моделей и обладающих известной спецификой:

– общегородские пространственно-временные «макромодели» геологии и гидрологии с учетом техногенной нагрузки, уточненные модели для проблемных зон (карст, оползни и т.п.);

– уникальные сооружения («небоскребы», мосты и туннели, над- и подземные торгово-культурно-развлекательно-спортивные комплексы, монументы и т.п.) и типовые жилые здания, в том числе, подверженные риску прогрессирующего обрушения (прежде всего, многоэтажные панельные);

– хранилища радиоактивных отходов, резервуары с нефтепродуктами и газами, дамбы и т.п.;

– системы «сооружение-основание» промышленных объектов, обеспечивающих нормальную жизнедеятельность мегаполиса и(или) влияющих на нее: атомные, тепловые и гидростанции, системы водоснабжения, опасные производства и др.;

– пространственные разветвленные трубопроводные системы промышленных производств (тепловая и атомная энергетика, химия и нефтепереработка и пр.), городских теплосетей и магистральных нефте- и газопроводов;

– транспортные внутри- и внешнегородские «узлы» и «артерии»: метро, автомагистрали, железные дороги, аэродромы и порты.

8.2.3. Понятие о программных комплексах компьютерного моделирования конструкций, зданий и сооружений.

Развитие любого численного или численно-аналитического метода расчета конструкций, зданий и сооружений обусловлено взаимосвязью трех основных факторов: наличие высокопроизводительной компьютерной техники (ЭВМ); разработка, совершенствование и развитие математических моделей исследуемых процессов и явлений (устойчивых и с достаточной степенью

точности адекватных реальным процессам и явлениям с достаточной степенью точности); особенности самого метода (в том числе в части адаптируемости к корректной алгоритмической реализации).

В основе подавляющего большинства современных универсальных и специализированных комплексов программ моделирования работы строительных конструкций лежит метод конечных элементов (МКЭ). В качестве характерных примеров можно указать, в частности, ABAQUS, ADYNA, ANSYS, DIANA, MicroFE, NASTRAN, Robot Millenium, SCAD, Лира, СТАДИО, отличающиеся все более возрастающей степенью автоматизации генерирования сети конечных элементов, формирования и решения огромного числа алгебраических уравнений, а также эффектными и эффективными средствами задания исходных данных и визуализации результатов (препроцессор и постпроцессор). У каждого конкретного комплекса программ есть свои сильные и слабые стороны, выбор пользователя зависит от его подготовленности в своей научной области, типа доступной ЭВМ, типа решаемой задачи, размерности решаемой задачи и других факторов. К значимым критериям, позволяющим сделать взвешенный выбор комплекса программ относятся: широта области применения, наличие реализации передовых научных достижений, коммерческая доступность, наличие подробной и понятной справочной документации.

8.3. Общая характеристика программного комплекса ЛИРА-САПР

8.3.1. Общая информация.

Программный комплекс (ПК) ЛИРА-САПР – это многофункциональный программный комплекс для расчета, исследования и проектирования конструкций, зданий и сооружений различного назначения. Разработчиком данного ПК является Общество с ограниченной ответственностью «Лира Сапр»

(Украина, г. Киев; официальный интернет-сайт компании: http://www.liraland.ru).

Помимо общего расчета модели объекта на различные виды статических (силовых, деформационных) и динамических (ветер с учетом пульсации, сейсмические воздействия по различным нормам, гармонические колебания и т.п.) нагрузок и воздействий ПК ЛИРА-САПР автоматизирует ряд процессов проектирования, среди которых, в частности, определение расчетных сочетаний нагрузок и усилий, назначение конструктивных элементов, подбор и проверка сечений стальных и железобетонных конструкций с формированием эскизов рабочих чертежей колонн и балок. Кроме того, в комплексе имеются возможности исследовать общую устойчивость рассчитываемой модели, проверить прочность сечений элементов по различным теориям разрушения, провести расчеты строительных объектов с учетом физической, геометрической и конструктивной нелинейностей, моделировать процесс возведения сооружения с учетом монтажа-демонтажа элементов с отслеживанием изменений физических свойств материалов.

8.3.2. Состав программного комплекса ЛИРА-САПР.

ПК ЛИРА-САПР состоит из нескольких взаимосвязанных информационных систем, обеспечивающих высокую технологичность работы с комплексом, начиная с этапа создания расчетной модели, и заканчивая вопросами к конструированию элементов.

ВИЗОР-САПР – это основная графическая система, единая графическая среда, располагающая обширным набором возможностей и функций для формирования адекватных конечноэлементных и суперэлементных моделей рассчитываемых объектов, их анализа и корректировки, описания физико-механических свойств матераилов, задания нагрузок и воздействий (в том числе взаимосвязей между различными загружениями с целью определения их наиболее опасных сочетаний), наглядной визуализации (на основе

высокопроизводительных графических решений) параметров напряженно-деформированного состояния (изополя перемещений и напряжений, эпюры усилий и прогибов), оторбражения мозаики разрушения элементов, главных и эквивалентных напряжений, форм потери устойчивости, анимации колебаний конструкции и др.

САПФИР – КОНСТРУКЦИИ – это так называемый архитектурный препроцессор, реализующий цепочку «Архитектурная модель – Аналитическая модель – Расчетная схема», который позволяет пользователю задавать исходную информацию, оперируя такими конструктивными элементами как плита, диафрагма, колонна, лестница, пандус и др.

Расчетные процессоры (солверы), входящие в состав ПК ЛИРА-САПР и реализующие современные высокопроизводительные методы решения систем уравнений большой вычислительной размерности, используются для расчета созданной модели (т.е. определения напряженно-деформированного состояния конструкций), сформированной на основе метода конечных элементов в перемещениях. Для решения линейно-упругих задач расчета конструкций используется линейный солвер, тогда как нелинейный солвер (на основе шаговых методов с автоматическим выбором шага нагружения и возможностью учета истории нагружения и шагово-итерационных методов) позволяет решать задачи в постановках (в том числе упруго-пластических), учитывающих физические нелинейности (свойства материалов (бетон, железобетон, сталебетон, металл, грунт)), геометричекие нелинейности (характерные для вантовых конструкций, большепролетных покрытий, мембран и др.) и конструктивные нелинейности (учет контактных взаимодействий, односторонних связей и трения), причем возможен одновременный учет нескольких видов нелинейностей.

Библиотека конечных элементов, реализованных в ПК ЛИРА-САПР, достаточно представительна и позволяет создавать адекватные расчетные модели практически без ограничений на описание реальных свойств

рассчитываемых объектов. Имеются возможности задания линейных и нелинейных законов деформирования материалов (в том числе реализаны законы деформирования различных классов железобетона), учета геометрической и конструктивной нелинейностей. Допускается наличие абсолютно жестких вставок в одномерных (стержневых) и двумерных (плоскостных) конечных элементах.

Вспомогательные расчетные процессоры призваны упростить и автоматизировать работу расчетчика и позволяют проводить дальнейшие исследования расчетной модели по результатам основного расчета. *Система РСУ* позволяет произвести выбор наиболее опасных сочетаний усилий по критерию экстремальных напряжений и в соответствии с нормативными национальными требованиями многих стран. *Система РСН* позволяет определить перемещения, усилия и напряжения от стандартных и произвольных линейных комбинаций загружений, причем под стандартными линейными комбинациями понимаются комбинации (сочетания), определенные соответствующими нормативными документами. *Система УСТОЙЧИВОСТЬ* дает возможность выполнить проверку общей устойчивости рассчитываемого объекта с определением коэффициента запаса и формы потери устойчивости. *Система ЛИТЕРА* реализует вычисление главных и эквивалентных напряжений по различным теориям прочности. *Система ФРАГМЕНТ* позволяет определить силы воздействия одного фрагмента рассчитываемого сооружения на другой как нагрузку (так, например, могут быть определены нагрузки, передаваемые наземной частью расчетной схемы на фундаменты). *Процессор Вариации моделей* предоставляет возможность комбинировать результаты расчета топологически идентичных расчетных схем, варьируя граничные условия, жесткостные характеристики, параметры упругого основания, жесткости узлов и т.п. *Система КС-САПР* (конструктор стандартных сечений) и *система КТС-САПР* (конструктор тонкостенных сечений) представляют собой специализированные графические среды для

формирования сечений произвольной конфигурации, снабженные расчетными модулями для определения осевых, изгибных, крутильных и сдвиговых характеристик, секториальных характеристик, координат центров изгиба и кручения, моментов сопротивления и формы ядра сечения, а при наличии усилий в заданном сечении производится визуалицяи текущих, главных и эквивалентных напряжений (соответствующих различным теориям прочности), эпюр секториальных характеристик.

В ПК ЛИРА-САПР реализованы возможности автоматизации конструирование стальных и железобетонных элементов рассчитываемого объекта по результатам комплекса проведенных основных и вспомогательных расчетов. Конструирующая *система АРМ-САПР* позволяет выполнить подбор площадей сечения арматуры колонн, балок, плит и оболочек по первому и второму предельным состояниям в соответствии с нормативами стран СНГ, Европы и США, в том числе с заданием произвольных характеристик бетона и арматуры (что особенно важно при расчетном обосновании на этапах реконструкции строительных объектов), имеются возможности объединения нескольких однотипных элементов в конструктивный элемент с увязкой арматуры по всей длине последнего. Допускается функинирование системы в локальном режиме (*ЛАРМ-САПР*) с подбором арматуры и проверкой заданного армирования для одного элемента, причем по результатам соответствующих расчетов в автоматизированном режиме формируются чертежи балок и колонн, производится создание выходных файлов чертежей. Конструирующая *система СТК-САПР* может работать в двух режимах – подбор сечений элементов стальных конструкций (таких как фермы, колонны и балки), и проверка заданных сечений в соответствии с нормативами стран СНГ, Европы и США; также допускается объединение нескольких однотипных элементов в конструктивный элемент, а сама система может функционировать в локальном режиме, позволяя проверить несколько вариантов при конструировании рассматриваемого элемента. *Система РС-САПР*, информационно связанная с

системой СТК-САПР, позволяет редактировать используемую сортаментную базу прокатных и сварных профилей. *Система ДОКУМЕНТАТОР* осуществляет формирование отчетов по результатам работы с комплексом, обеспечивая наглядное представление (в том числе экспорт в офисные приложения) всей необходимой информации в графическом и текстовом (табличном, в том числе с возможностью последующей обработки данных в программе *Дизайнер таблиц*) видах.

На базе ПК ЛИРА-САПР разработано также несколько специализированных расчетно-графических систем. *Система МОНТАЖ-плюс* реализует моделирование работы сооружения в процессе возведения при многократном изменении расчетной схемы и позволяет выполнять моделирование возведения высотных зданий из монолитного железобетона с учетом изменений жесткости и прочности бетона, вызванных временным замораживанием уложенной смеси и иными факторами. *Система МОСТ* дает возможность произвести построение поверхностей и линий влияния в мостовых сооружениях от подвижной нагрузки. *Система ДИНАМИКА-плюс* реализует метод прямого интегрирования уравнений движения по времени, обеспечивая адекватное компьютерное моделирование вынужденных колебаний физически и геометрически нелинейных систем. *Система КМ-САПР* позволяет получить полный комплект чертежей КМ в среде AutoCAD [233] (монтажные схемы с маркировкой элементов и узлов, ведомости элементов, чертежи узлов с трехмерной визуализацией, а также их спецификации) на основании результатов расчета стальных конструкций (элементов и узлов). *Система САПФИР–ЖБК* обеспечивает формирование в автоматизированном режиме рабочих чертежей армирования плит и диафрагм (раскладка арматуры, спецификации, ведомости деталей и др.) по результатам подбора арматуры в плитах перекрытий и диафрагмах. *Система ГРУНТ* выполняет построение трехмерной модели грунтового массива по данным инженерно-геологических изысканий (положение и характеристики скважин), а также реализует

определение коэффициентов постели в каждой точке проектируемой фундаментной плиты.

ПК ЛИРА-САПР позволяет использовать различные действующие системы единиц измерения, как на этапе создания модели, так и при анализе результатов расчета.

8.4. Общая характеристика программного комплекса SCAD Office

8.4.1. Общая информация.

Единая графическая среда создания расчетной схемы, выполнения расчетов и анализа результатов в комплексе SCAD Office обеспечивает широкие возможности моделирования, как для самых простых конструкций, так и для достаточно сложных зданий и сооружений, удовлетворяя при этом потребностям различных групп пользователей. Высокопроизводительный солвер позволяет решать задачи большой размерности (сотни тысяч степеней свободы при статических и динамических воздействиях). Разработчиком данного ПК является компания SCAD Soft (Украина, г. Киев; официальный интернет-сайт компании: http://scadsoft.com/).

В ПК SCAD Office имеются: развитая библиотека конечных элементов для моделирования стержневых, пластинчатых, твердотельных и комбинированных конструкций; модули анализа устойчивости, формирования расчетных сочетаний усилий, проверки напряженного состояния элементов конструкций по различным теориям прочности, определения усилий взаимодействия фрагмента с остальной конструкцией, вычисления усилий и перемещений от комбинаций загружений. В состав комплекса включены программы подбора арматуры в элементах железобетонных конструкций и проверки сечений элементов металлических конструкций. В ПК SCAD Office постоянно развивается, совершенствуются интерфейс пользователя и вычислительные возможности, создаются новые проектирующие компоненты.

В состав ПК SCAD Office входят программы нескольких видов:

Вычислительный комплекс Structure CAD (SCAD), являющийся универсальной расчетной системой конечно-элементного анализа конструкций и ориентированный на решение задач проектирования зданий и сооружений достаточно сложной структуры;

Вспомогательные программы, предназначенные для «обслуживания» SCAD и обеспечивающие форматирование и расчет геометрических характеристик различного вида сечений стержневых элементов (Конструктор сечений, КОНСУЛ, ТОНУС, СЕЗАМ), определение нагрузок и воздействий на проектируемое сооружение (ВеСТ), вычисление коэффициентов постели, ли, необходимых при расчете конструкций на упругом основании (КРОСС), импорт данных из архитектурных систем и формирование укрупненных моделей (препроцессор ФОРУМ);

Проектно-аналитические программы (КРИСТАЛЛ, АРБАТ, ЗАПРОС, ДЕКОР, КАМИН, ОТКОС), предназначенные для решения частных задач проверки и расчета стальных и железобетонных конструкций в соответствии с требованиями нормативных документов (СНиП, СП), расчета элементов оснований и фундаментов, расчетов и проверок элементов каменных и армокаменных конструкций на соответствие требованиям СНиП;

Проектно-конструкторские программы (КОМЕТА, МОНОЛИТ), предназначенные для разработки конструкторской документации на стадии детальной проработки проектного решения;

Электронные справочники (КоКон, КУСТ).

8.5. Общая характеристика программного комплекса СТАДИО

8.5.1. Общая информация.

Многоцелевой ПК СТАДИО обеспечивает высокоточное численное решение стационарных и нестационарных задач теории поля, определение

статического, температурного и динамического напряженно-деформированного состояния объекта исследования, выполняет анализ устойчивости и прочности произвольных комбинированных механических систем (массивно-оболочечно-пластинчато-мембранно-стержневых с «жесткими» телами, полостями жидкости и внутренними связями, с изо- и ортотропными материалами) в плоской, осесимметричной и трехмерной линейной и нелинейной постановках. Разработчиком данного ПК является Закрытое акционерное общество «Научно-исследовательский центр СтаДиО» (Россия, г. Москва; официальный интернет-сайт компании: http://stadyo.ru).

В комплексе с развитым сервисным обеспечением имеется обширная библиотека прямолинейных и криволинейных стержневых, изгибно-мембранных, криволинейных супер- и изо-параметрических конечных элементов (в общей сложности более 120 типов), а также элементов с заданными матрицами жесткости, инерции и демпфирования. Для проведения расчетов используются современные оптимизированные солверы, реализующие соответственно метод Холецкого, метод сопряженных градиентов с предобуславливанием (PCG), блочный метод Ланцоша, метод итераций подпространства, явные и неявные разностные схемы интегрирования (центральных разностей, Ньюмарка, Вилсона и Кранка-Николсона), суперэлементные схемы (иерархия и степень «вложенности» подконструкций (для суперэлементной «ветви») – произвольные) и др. Принципиальный шаг сделан в уточненном учете динамических характеристик сложных подсистем в составе общей модели (формирование и использование редуцированных матриц влияния Крейга-Бемптона).

8.6. Общая характеристика программного комплекса АСТРА-НОВА

8.6.1. Общая информация.

Семейство программных комплексов АСТРА-НОВА, аттестованных в Госатомнадзоре РФ, обеспечивает автоматизированные расчеты произвольных пространственных разветвленных и протяженных, низко- и высокотемпературных, надземных (на опорно-подвесной системе), наземных и подземных трубопроводных систем на статическую и циклическую прочность, на сейсмические воздействия, вибропрочность и неустановившиеся динамические процессы в соответствии с требованиями действующих российских нормативных документов.

Разработчиком данного ПК является Закрытое акционерное общество «Научно-исследовательский центр СтаДиО» (Россия, г. Москва; официальный интернет-сайт компании: http://stadyo.ru).

В комплексах реализован единый алгоритм расчета трубопроводов (определение перемещений, нагрузок на опоры и усилий в сечениях) как линейно-упругих пространственных многократно статически неопределимых стержневых систем, сочетающий суперэлементный подход метода перемещений, методы начальных параметров и прогонки (для каждого суперэлемента) и спектральную методику решения динамических задач. Учитывается повышенная оболочечная податливость криволинейных труб (эффекты Кармана), тройниковых соединений и штуцерных врезок. Расчет подземных трубопроводов с бесканальной прокладкой опирается на результаты численных и экспериментальных исследований трехмерных нелинейных систем «трубопровод-изоляция-грунт». Значимые собственные частоты и формы колебаний в требуемом частотном диапазоне определяются из решения частной проблемы собственных значений методом блочного Ланцоша, а динамические расчеты (сейсмические, вибрационные и на неустановившуюся динамику) выполняются спектральным методом с суперпозицией реакций по собственным формам колебаний. Принципиальный шаг сделан в уточненном учете статических и динамических характеристик сложных подсистем (детали, опорные конструкции и оборудование) в составе общей суперэлементной

модели трубопроводной системы – формирование и использование так называемых редуцированных матриц влияния суперэлементов (Крейга-Бемптона, жесткости, масс, нагрузок).

В целом, впервые в отечественной практике достигнут качественно новый уровень комплексного автоматизированного расчетного обоснования статической и циклической прочности, сейсмостойкости, вибрационной и динамической прочности на доступных компьютерах: трубопроводные системы произвольной сложности можно (и следует) оперативно и точно моделировать с использованием преимуществ современных численных методов, Windows- и САПР-технологий, анализировать в полном соответствии с требованиями действующих российских норм и оптимизировать по прочностным критериям, не прибегая к вынужденным и, зачастую, необоснованным упрощениям и умолчаниям.

8.6.2. Состав программного комплекса АСТРА-НОВА.

Любой отраслевой комплекс семейства АСТРА-НОВА состоит из перечисленных ниже программных модулей:

ПРЕ-АСТРА – многофункциональный препроцессор комплекса, учитывающий разнообразные «традиции» различных отраслей, имеющий диалоговые многооконные режимы задания, генерацию и визуализацию рациональных пространственно-стержневых расчетных моделей трубопроводных систем (с возможностями использования баз данных по материалам, опорам, пружинным подвескам и фланцевым соединениям, ранее заданных моделей, расстановки опор и распределение динамических степеней свободы). Реализована совместимость с предыдущими версиями АСТРА-НОВА, другими программами расчета трубопроводов (СТАРТ, CAESAR II) и универсальными расчетными ПК (ANSYS, СТАДИО), двусторонний интерфейс с САПР, ориентированными на использование в строительной отрасли

(CADWorx/PIPE, SmartPlant, Plant 3D, …), поиск и отображение коллизий (пересечений, касаний и недопустимо малых зазоров труб). Пользователь может визуализировать компоновку трубопроводной системы, оперируя понятиями «прямая труба», «отвод», «тройник», «опора», «пружина» и др., используя базы данных и ранее заданные проекты. Оптимальная расчетная суперэлементная модель трубопровода генерируется автоматически с возможностью дальнейшей корректировки пользователем.

АСТРА-ДЕТАЛЬ – расчет по выбору основных размеров деталей в соответствии с требованиями отраслевых российских норм. Реализован расчёт деталей трубопроводов – прямых труб, отводов (гибов), переходов, тройников и крышек/днищ – на действие давления. Предусмотрено два вида расчета: выбор минимальной расчетной и номинальной (с учетом прибавок) толщины стенки деталей; выбор деталей, подходящих по толщине стенки, из сортаментов.

АСТРА-СТАЦ – расчет на статическую и циклическую прочность низко- и высокотемпературных трубопроводов. Реализованы режимы задания или рационального выбора характеристик пружинных подвесок из сортамента МВН, ОСТ, LISEGA, «спецпружин», пружин постоянного усилия и пружин пользователя. Предусмотрены все практически значимые виды прочностных расчетов (в том числе, с учетом трения в опорах по двум альтернативным методикам) на нормативно регламентируемые сочетания квазистатических и малоцикловых нагрузок для всех режимов эксплуатации. Для тяжело нагруженных отводов (колен и гибов), тройников (врезок) и линзовых компенсаторов выполняется расчет напряжений и прочности по уточненным методикам. Дополнительно допускается задание опор в местной системе координат и стержневых элементов произвольного сечения (для моделирования, например, сложных опорных конструкций совместно с трубопроводом), учет отрыва трубопровода от опор. Для АСТРА-ТЕПЛОСЕТЬ и -МАГИСТР реализованы уточненные алгоритмы расчета подземных

трубопроводов с бесканальной прокладкой, аналогичные используемым в Еврокодах.

АСТРА-ФОРМ – определение значимых собственных частот и форм колебаний системы в заданном частотном диапазоне (модуль необходим и как «препроцессор» программ динамического расчета – -СЕЙСМ, -ВИБР и -ДИН), в том числе, с учетом статического «статуса» опор односторонних и (или) трения.

АСТРА-СЕЙСМ – расчет на сейсмические воздействия, заданные одно- или многокомпонентными спектрами ответа (линейно-спектральный подход) или ответными акселерограммами (интегрирование по времени) на отметках крепления, с учетом всех значимых собственных частот и форм колебаний системы, а также вклада высших форм. Реализованы также расчет на «высокочастотное» воздействие и автоматизированный режим рациональной расстановки и учета сейсмоопор. Результаты: сейсмические перемещения, ускорения, усилия, нагрузки и напряжения, оценка сейсмостойкости.

АСТРА-ВИБР – расчет на вибропрочность для режимов установившихся вынужденных колебаний. Предусмотрено два вида расчета: определение допускаемых амплитуд (нормировка) виброперемещений по каждой учитываемой собственной форме и оценка амплитуд виброперемещений, напряжений и нагрузок, вибропрочности и долговечности при заданных гидродинамических нагрузках.

АСТРА-ДИН – расчет неустановившихся вынужденных колебаний при заданных динамических нагрузках для различных переходных и аварийных режимов эксплуатации (открытие-закрытие клапанов, пропуск снарядов и т.п.).

ПОСТ-АСТРА – «постпроцессор», включающий выдачу развернутых и компактных выборочных таблиц результатов статических и динамических расчетов (на русском и/или английском языках), визуализацию расчетной модели, перемещений, нагрузок на опоры, усилий и напряжений в элементах, коллизий в деформированных состояниях, анимацию собственных форм и

вынужденных колебаний с возможностью экспорта результатов (в том числе в форме сгенерированных отчетов) в офисные приложения и CAD-системы.

АСТРА-СТАДИО – уточненный расчет температурного и напряженно-деформированного состояний, статической, сейсмической и циклической прочности типовых деталей-элементов трубопроводов: сварных, с накладками, наплавкой и штампованных тройников, отводов (гибов, колен, секторных) с учетом влияния присоединенных труб, эллиптичности и разностенности, конических концентрических и эксцентрических переходов, линзовых и сильфонных компенсаторов, сварных соединений, труб с изоляцией в грунте и т.п. Подсистема обладает собственным развитым диалоговым пре- и постпроцессором, связана с базой данных типовых элементов, требует лишь необходимого минимума информации для генерации оптимальных пространственно-оболочечных и трехмерных расчетных моделей; вычислительным «ядром» служит мощный конечно-элементный комплекс СТАДИО, рассмотренный ранее в параграфе 4.4 настоящего учебного пособия.

АСТРА-РАЗР – расчет последствий мгновенного аварийного разрыва трубопровода, в том числе – пространственно-временное распределение гидродинамических нагрузок, движение трубопроводов с учетом больших перемещений и пластических деформаций, оценка ударных нагрузок на опоры-ограничители и близлежащие системы.

8.7. Общая характеристика программного комплекса MicroFe

8.7.1. Общая информация.

MicroFe – это еще один программный комплекс конечноэлементных расчетов пространственных конструкций на прочность, устойчивость и колебания. Разработчиками данного ПК являются Общество с ограниченной ответственностью «Техсофт» (Россия, г. Москва; официальный интернет-сайт

компании: http://www.tech-soft.ru) и фирма mb AEC Software GmbH (Германия, г. Кайзерслаутерн).

ПК MicroFe предоставляет пользователям широкие возможности решения статических и динамических (собственные колебания, расчеты на динамические воздействия) задач в линейной и нелинейной постановках (в том числе с учетом нелинейных связей), проводить анализ устойчивости (в том числе с учетом физической нелинейности), выполнять комплексные исследования работы конструкции. Кроме того, также предусмотрен ряд дополнительных видов расчетов, среди которых, в частности, следующие: расчет на прогрессирующее разрушение, решение задачи идентификации, индикация погрешностей, определение спектральных свойств матрицы жесткости (это позволяет выявить слабые места конструкции и содействует поиску оптимального расположения и сечений элементов несущих конструкций).

8.7.2. Особенности программного комплекса MicroFe.

Некоторые важные особенности ПК MicroFe перечислены ниже.

– в библиотеку конечных элементов комплекса входя *гибридные конечные элементы*, обеспечивающие при решении определенного круга задач высокую точность расчета без использования излишне подробных конечноэлементных сеток;

– формирование модели ведется *в понятных инженеру-строителю терминах* (в качестве составляющих частей модели фигурируют обычные строительные элементы (плита, стена, колонна, балка и др.)), реализованы развитые возможности построения модели, использования информации о модели из специализированных архитектурных и графических программ;

– реализованы *специальные инструменты для корректного учета стыков строительных конструкций* типа колонна – плита, балка – стена, стена – плита, плита – ребро, позволяющие корректно смоделировать соответствующие реальные связи и получить корректные результаты для перечисленных типов

стыков без дополнительных затрат труда (большинство из соответствующих инструментов могут быть сгенерированы автоматически), причем работа с несогласованными сетками позволяет получить качественные конечно-элементные сетки при корректном моделировании стыков конструктивных элементов;

– реализована *модель слоистого грунтового основания с возможностью задания нелинейных свойств соединения фундаментов с грунтовым массивом и нелинейных свойств грунта* позволяет корректно учесть влияние работы основания на несущую конструкцию. Модель учитывает различные свойства по слоям, влияние соседних строений, действие нагрузки от собственного веса грунта, что невозможно при использовании параметрических моделей упругого основания. При работе со слоистым основанием могут быть рассмотрены задачи со свайно-плитными фундаментами, с учетом нелинейных свойств грунта и связи грунта и сваи;

– мощное расчетное ядро позволяет решать задачи большой размерности за относительно короткое время даже на обычных персональных компьютерах, причем *автоматическое распараллеливание вычислений* дает возможность использовать все ресурсы многопроцессорных (многоядерных) компьютеров для ускорения расчетов.

– предусмотрено *выполнение конструктивных расчетов с применением понятия «конструктивный элемент»* для железобетонных и стальных конструкций, что делает задание данных для конструктивных расчетов и анализ результатов простым и понятным, при этом реализованы возможности автоматического преобразования позиций (строительных элементов) в конструктивные элементы, имеется развитый инструментарий для редактирования групп и элементов, хранения результатов, что, очевидно, облегчает работу инженера-конструктора;

– в составе комплексе *оперативно реализуются актуальные версии нормативных документов*;

– предусмотрены возможности проведения *целого комплекса специализированных расчетов*, среди которых, в частности следующие: расчеты на прогрессирующее обрушение; расчет теплопроводности; расчет на сейсмическое (динамическое) воздействие с учетом работы нелинейных связей (сейсмоизоляторов); оценка надежности железобетонных стержневых конструкций вероятностными методами; учет этапности возведения с возможностью просмотра результатов по каждому этапу (т.е. возможно моделирование работы конструкции с учетом технологии и последовательности возведения (в том числе для нелинейных задач));

– обеспечена связь с другими программами проектирующей системы ING+ (еще одна разработка ООО «ТЕХСОФТ») и программами сторонних производителей программного обеспечения, позволяющая выстроить сквозную технологию проектирования строительных конструкций.

8.8. Общая характеристика программных комплексов Midas Civil и Midas GTS

8.8.1. Общая информация.

Midas CIVIL – это верифицированный в системе РААСН конечно-элементный программный комплекс, предназначенный, прежде всего, для расчета и проектирования мостов и транспортных сооружений (вместе с тем, разумеется, также возможно проведение расчетов конструкций общего вида). Разработчиком данного ПК является фирма MIDAS Information Technology Co., Ltd. /MIDAS IT/ (Южная Корея, г. Сеул; официальный интернет-сайт компании: http://www.midasit.ru).

ПК Midas CIVIL позволяет проводить следующие типы расчетов рассматриваемых объектов: линейный статический расчет, определение температурных напряжений, линейный динамический расчет, определение собственных частот и форм колебаний, расчет спектра отклика, моделирование

поведения конструкции во времени, линейный расчет устойчивости, нелинейные статические расчеты, расчет P-Delta, расчет в условиях больших перемещений, нелинейный расчет в случае нелинейных элементов, расчет предельного равновесия, расчет на подвижную нагрузку.

Midas GTS – это еще один верифицированный в системе РААСН программный комплекс, предназначенный для комплексных геотехнических расчетов, обеспечиывющий комплексное решение актуальных проблем расчета и проектирования в геотехническом строительстве и тоннелестроении. Разработчиком данного ПК является фирма MIDAS Information Technology Co., Ltd. /MIDAS IT/ (Южная Корея, г. Сеул; официальный интернет-сайт компании: http://www.midasit.ru).

Области применения Midas GTS: расчет тоннелей с учетом сложных условий взаимодействия с окружающей средой; расчет котлованов и временных конструкций (глубокие котлованы для оснований наземных конструкций, расчет временных конструкций с учетом уже существующих сооружений, например, при расчете конструкций метрополитена; расчет фильтрации грунтовых вод; расчет конструкций обделки; расчет насыпи на мягком грунте и консолидации грунта; расчет любых конструкций, соприкасающихся с грунтом (дамбы, плотины, ж.д. насыпи, волнорезы, причалы, основания зданий и сооружений и др.).

8.9. Общая характеристика программного комплекса MSC Nastran

8.9.1. Общая информация.

MSC Nastran – это универсальный многоцелевой конечно-элементный программный комплекс (ПК) мирового уровня, так называемый «тяжелый» конечно-элементный программный комплекс. Разработчиком данного ПК является фирма MSC. Software Corporation (США, Калифорния; официальный интернет-сайт компании: http://www.mscsoftware.ru).

MSC Nastran обеспечивает полный набор наиболее востребованных типов расчетов, включая определение напряженно-деформированного состояния; расчет запасов прочности; определение собственных частот и форм колебаний; анализ устойчивости; исследование установившихся и неустановившихся динамических процессов; решение задач теплопередачи, акустических явлений, нелинейных статических и нелинейных переходных процессов; расчет с учетом сложного контактного взаимодействия; расчет критических частот и вибраций роторных машин; анализ частотных характеристик при воздействии случайных нагрузок и импульсного широкополосного воздействия; исследование конструкции объекта с учетом аэроупругости на дозвуковых и сверхзвуковых скоростях. Предусмотрена возможность моделирования практически всех типов материалов, включая композитные и гиперупругие. В состав расширенных функций входит технология суперэлементов (подконструкций), включая продвинутые методы динамических конденсаций, модальный синтез и развитые методы анализа динамики сложных структур на основе суперэлементов и формулировок метода Крейга-Бемптона.

Комплекс имеет многоуровневую структуру. Программные модули нижнего уровня разработаны преимущественно с использованием алгоритмического языка Fortran. Модули верхнего уровня построены с использованием внутреннего языка Direct Matrix Abstraction Program (DMAP). Пользователи MSC Nastran могут разрабатывать свои проблемно-ориентированные модули на языке DMAP и включать их в MSC Nastran, создавая новые собственные решения. Еще более высокий уровень – это внедрение в структуру MSC Nastran технологии Service Component Architecture (SCA), предназначенной для интеграции специально разработанных пользователем программных компонентов в расчетную конечноэлементную модель изделия, что существенно расширяет возможности учета «тонких» особенностей моделируемого изделия.

8.9.2. Особенности программного комплекса MSC Nastran.

В MSC Nastran предусмотрена возможность моделирования практически всех типов материалов, в том числе изотропных, ортотропных, анизотропных, материалов с температурно-зависимыми характеристиками, гиперупругих материалов и композиционных материалов.

Предусмотрена возможность использования практически всех типов конечных элементов, включая элементы высокого порядка аппроксимации (P-элементы), которые хорошо отражают криволинейную геометрию конструкции, автоматически адаптируются к желаемому уровню точности и обеспечивают высокую точность при детальном расчете напряжений. В числе реализованных в комплексе элементов следует отметить следующие: скалярные и специальные элементы; одномерные балочные элементы; двумерные оболочечные элементы; трехмерные объемные элементы; элементы демпфирования; нелинейные элементы для анализа статических и переходных процессов; «жесткие» элементы (кинематические межузловые связи); элементы для моделирования точечной и шовной сварки, болтовых и заклепочных соединений; специальные элементы для акустических расчетов; P-элементы высокого порядка аппроксимации; элементы-интерполяторы; элементы для моделирования композитных конструкций; осесимметричные элементы; вершина трещины; элементы общего назначения; элемент-коннектор; суперэлементы (подконструкции) и др.

В состав расширенных функций MSC Nastran входит технология суперэлементов (подконструкций), включающая продвинутые методы динамической конденсации, модальный синтез и развитые методы анализа динамики сложных структур на основе суперэлементов и формулировок метода Крейга–Бамптона. Применение технологии суперэлементов позволяет выполнять расчёты с использованием конечно-элементных моделей, размерность которых превосходит возможности имеющейся вычислительной техники, а также существенно сокращать трудозатраты и затраты времени при

исследовании влияния локальных модификаций конструкции на характеристики изделия в сборе.

В MSC Nastran реализован эффективный аппарат оптимизации изделий для задач статики, устойчивости, установившихся и неустановившихся динамических процессов, собственных частот и форм колебаний, акустики и аэроупругости. Оптимизация проводится на основе выбранных типов расчета путем вариации параметров формы, размеров и свойств конструкции. Благодаря своей эффективности алгоритмы оптимизации обрабатывают неограниченное число проектных параметров и ограничений. Масса, напряжения, перемещения, частоты собственных колебаний и многие другие характеристики могут рассматриваться либо в качестве целевых функций проекта (в этом случае их можно минимизировать или максимизировать), либо в качестве ограничений. Алгоритмы анализа чувствительности позволяют исследовать влияние различных параметров на поведение целевой функции и управлять процессом поиска оптимального решения.

Широкие возможности оптимизации позволяют использовать MSC Nastran для автоматической идентификации компьютерной расчетной модели, при этом целевая функция, подлежащая минимизации, определяется как величина рассогласования результатов расчета и эксперимента, а в качестве варьируемых параметров выбираются наименее достоверные расчетные параметры конструкции. Результат решения такой задачи – это «новая» конечноэлементная модель, свойства которой соответствуют свойствам физического образца.

Интеграция в MSC Nastran решателей, основанных на разных математических принципах (неявных и явных решателей) позволяет с высокой эффективностью моделировать процессы последовательных этапов функционирования изделий, характеризующихся существенно разными характерами протекающих процессов.

8.10. Общая характеристика программного комплекса ANSYS

8.10.1. Общая информация.

ANSYS – это универсальный многоцелевой конечноэлементный программный комплекс (ПК) мирового уровня (так называемый «тяжелый» конечноэлементный программный комплекс), реализующий развитые схемы метода конечных элементов и метода суперэлементов для статических и динамических расчетов пространственных комбинированных систем. Разработчиком данного ПК является фирма ANSYS, Inc. (США, г. Канонсберг; официальный интернет-сайт компании: http://www.ansys.com). Заметим, что под общим названием ПК ANSYS нередко понимают целое семейство программных продуктов компании ANSYS, Inc.

Каждая версия ПК ANSYS включает новые и расширяет прежние возможности программного комплекса, делая последний более быстродействующим, более гибким и удобным в использовании. Несмотря на то, что ПК ANSYS располагает богатыми и сложными возможностями, организационная структура комплекса и «дружеский» графический интерфейс пользователя делают изучение и применение данного ПК очень удобным. ПК ANSYS распространен в Российской Федерации и в мире, имеет более полумиллиона легальных пользователей.

8.10.2. Приложения программного комплекса ANSYS.

В целом, все программные продукты ANSYS могут быть классифицированы на основе областей приложений, на которые они ориентированы.

Механика дефомируемого твердого тела. Одной из основных задач при проектировании изделий (конструкций, зданий и сооружений) является обеспечение прочности и надежности изделия при эксплуатационных нагрузках. Компания ANSYS, Inc. представляет широкий спектр решений для

определения напряженно-деформированного состояния конструкций, динамического анализа, оценки температурного состояния узлов и выполнения связанных расчетов. Эти возможности комплекса в разной мере представлены в комплексах ANSYS Mechanical, ANSYS Structural, ANSYS Professional NLS, ANSYS Professional NLT и ANSYS DesignSpace.

ANSYS Mechanical позволяет решать широкий спектр задач механики деформируемого твердого тела и строительной механики с учетом нелинейных свойств материалов, пластичности и наличия контактного взаимодействия. Кроме того, могут быть корректно решены задачи линейной / нелинейной динамики, теплообмена, акустики, имеются возможности решения связанных задач.

ANSYS Structural содержит все возможности, реализованные в ANSYS Mechanical за исключением тех, которые относятся к решению задач теплообмена.

ANSYS Professional содержит большинство возможностей, реализованных в ANSYS Mechanical, однако обладает несколько ограниченным функционалом при решении задач с нелинейностями.

ANSYS DesignSpace является простым и удобным младшим продуктом в линейке программных продуктов для решения прочностных задач ANSYS; содержит базовые возможности ANSYS Mechanical для решения прочностных и тепловых задач.

Быстропротекающие высоконелинейные динамические процессы. Для моделирования быстропротекающих процессов и решения задач с большими деформациями и напряжениями на основе использования явных методов, компаней ANSYS, Inc. были разработаны программные продукты ANSYS LS-DYNA, ANSYS AUTODYN и ANSYS Explicit STR.

ANSYS LS-DYNA – программный комплекс для анализа высоконелинейных динамических процессов, объединяющий возможности мощной пре- и постпроцессорной обработки программного обеспечения от ANSYS, Inc.,

параметрического языка программирования ANSYS APDL и явного динамического решателя LS-DYNA, разработанного компанией Livermore Software Technology Corporation (LSTC).

ANSYS AUTODYN – один из лучших программных комплексов для решения задач нелинейной динамики сплошных сред, жидкостей газов и их взаимодействия, получивщий большую популярность при решении задач из области физики взрывов и ударов, а также при определении отклика конструкций на ударно-волновое воздействие.

ANSYS Explicit STR позволяет решать задачи нестационарной нелинейной динамики явными методами в расчетном модуле Workbench Mechanical среды ANSYS Workbench (см. ниже) с использованием решателя ANSYS AUTODYN.

Вычислительная гидрогазодинамика. Для решения задач вычислительной гидрогазодинамики компания ANSYS, Inc. предлагает два CFD-пакета: ANSYS CFX и ANSYS FLUENT. Оба пакета, основанные на методе конечных объемов, обеспечивающем высокую точность получаемых результатов, содержат расширенные наборы моделей турбулентности, решателей, библиотеки материалов (жидкость / газ). Комплексы позволяют моделировать движение потоков жидкостей или газов в объектах с подвижными границами (клапаны, поршни и т.п.), а также в связке с ANSYS Mechanical решать задачи взаимодействия жидкости и твердого тела (Fluid – Structure Interaction (FSI)).

ANSYS CFD обеспечивает полный доступ ко всем функциональным возможностям CFD-кодов ANSYS: ANSYS CFX и ANSYS FLUENT. ANSYS CFD предназначен для моделирования ламинарных и турбулентных потоков (с учетом/без учета сжимаемости среды); расчета процессов теплообмена (конвекцией, теплопроводностью, излучением), процессов горения; моделирования многофазных потоков и решения задач акустики.

ANSYS CFD-Post – унифицированный постпроцессор для CFD продуктов компании ANSYS, Inc.

Электромагнетизм и проектирование электронных устройств. Электромагнитные решения, предназначенные для моделирования электромагнитных полей, ANSYS представлены несколькими продуктами: ANSYS Maxwell, ANSYS HFSS, ANSYS Emag (на базе ANSYS Mechanical APDL). Каждый из продуктов ориентирован на тот или иной сегмент рынка электронных устройств и электротехнических изделий, от мощных силовых трансформаторов до устройств типа МЭМС (микроэлектромеханические системы).

ANSYS Maxwell – специализированный программный комплекс для моделирования электромагнитных полей, который используется при проектировании таких устройств, как электромоторы, переключатели, трансформаторы и другие электронные устройства.

ANSYS HFSS предназначен для моделирования трехмерных высокочастотных электромагнитных полей, определения S-параметров, анализа электромагнитной совместимости устройств, построения частотно-зависимой модели схем распределения питания и пр.

ANSYS Simplorer – программный продукт для моделирования изделий на уровне системы.

ANSYS Icepak используется для определения температурного состояния электронных устройств и компонентов, содержит встроенный генератор гексаэдральных сеток со ступенчатой аппроксимацией и CFD-решатель ANSYS FLUENT, причем комплекс позволяет импортировать геометрию из MCAD-/ECAD-систем через форматы IDF, MCM, BRD, TCB – «электронные» и STEP, IGES, DXF – геометрические.

Многодисциплинарные расчеты в ANSYS. Многодисциплинарное моделирование встречается во многих приложениях, например, в биомедицине (волнообразные колебания эластичных стенок артерий), в авиастроении (флаттер), строительстве (ветровые и волновые нагрузки на сооружения),

трубопроводном транспорте (гидравлический удар и вибрации), электронике (электроупругость, термоэлектричество и пьезо-эффекты).

ANSYS Multiphysics – это наиболее полная комплектация расчетного комплекса ANSYS, позволяющая объединить задачи теплообмена, гидрогазодинамики, электромагнетизма, механики деформируемого твердого тела и строительной механики в одну комплексную многодисциплинарную задачу, решаемую с помощью всего диапазона мощных прямых и итерационных солверов (решателей) компании ANSYS, Inc.

ANSYS AIM — концептуально новая расчетная платформа ANSYS, ориентированная на проведение многодисциплинарных расчетов.

8.11. Общая характеристика программного комплекса SIMULIA Abaqus

8.11.1. Общая информация.

SIMULIA Abaqus – это универсальный многоцелевой конечноэлементный программный комплекс (ПК) мирового уровня (так называемый «тяжелый» конечноэлементный программный комплекс), который может эффективно использоваться для точного и достоверного решения различных инженерных задач, а также проведения фундаментальных и прикладных научных изысканий. Семейство продуктов Abaqus разрабатывается и поддерживается компанией Abaqus, Inc. (США) с 1978 года, а начиная с 2005 года Abaqus, Inc. входит в компанию Dassault Systemes (Франция, г. Марсель; официальный интернет-сайт компании: http://www.3ds.com).

В настоящее время ПК SIMULIA Abaqus находит широкое применение в таких отраслях промышленности как энергетика (в том числе в таких компаниях как ABB, AEA Technology, SIEMENS, EPRI и др.), автомобилестроение (в том числе в таких компаниях как BMW, FORD, General Motors, Mercedes, Toyota, Volvo, Goodyear), авиастроение и ВПК (в том числе в

таких компаниях как General Dynamics, Lockheed Martin, US Navy, Boeing), электроника (в том числе в таких компаниях как (Intel, Hewlett-Packard, Motorola, IBM, Digital), металлургия (в том числе в таких компаниях как British Steel, DuPont), нефтедобыча и переработка (в том числе в таких компаниях как Exxon/Mobil, Shell, Dow), производство товаров народного потребления (в том числе в таких компаниях как 3M, Kodak, Gillette) и других. ПК SIMULIA Abaqus используется в научно-образовательном процессе ведущих российских университетов, среди которых Московский государственный технический университет имени Н.Э. Баумана, Московский физико-технический институт (государственный университет), Южный федеральный университет, Санкт-Петербургский политехнический университет Петра Великого, Южно-Уральский государственный университет (национальный исследовательский университет), Национальный исследовательский Московский государственный строительный университет, Пермский национальный исследовательский политехнический университет и др.

8.11.2. Состав программного комплекса SIMULIA Abaqus.

ПК SIMULIA Abaqus разработан по модульному принципу. Он состоит из двух основных модулей – решателей (солверов) Abaqus/Standard и Abaqus/Explicit, пре-постпроцессора Abaqus/CAE и дополнительных модулей, учитывающих особенности специфических проблем (Abaqus/Aqua, Abaqus/Design, FE-Safe и др.). Все модули эффективно дополняют друг друга. Ниже приведена их очень краткая характеристика.

Abaqus/Standard – один из двух основных решателей программного комплекса SIMULIA Abaqus, использующий неявную формулировку метода конечных элементов. Abaqus/Standard позволяет использовать различные методы статического и динамического расчета строительных конструкций, зданий, сооружений и комлпексов.

Abaqus/Explicit – решатель для сильно нелинейных переходных быстротекущих динамических процессов, использующий явную схему интегрирования метода конечных элементов.

Abaqus/Design – дополнительный модуль к Abaqus/Standard, позволяющий анализировать чувствительность к изменению параметров конструкции и проводить оптимизацию.

ABAQUS/Aqua – дополнительный модуль к Abaqus/Standard, позволяющий анализировать нагрузки на кабели, трубопроводы и другие конструкции, погруженные в воду.

Abaqus/Electromagnetic – данный модель ориентирован на численное решение проблем электромагнетизма.

FE/Safe – данный модуль использует результаты расчета с использованием ПК SIMULIA Abaqus для анализа усталостной прочности, долговечности, ресурсоемкости конструкций.

Abaqus/ADAMS – данный интерфейс позволяет экспортировать результаты из ПК SIMULIA Abaqus в ADAMS/Flex.

Abaqus for CATIA – данный интерфейс позволяет готовить модели и просматривать результаты из ПК SIMULIA Abaqus непосредственно в CAD-системе CATIA (программный продукт компании Dassault Systemes).

Abaqus/MOLDFLOW – данный этот интерфейс транслирует информацию из набора средств моделирования процесса литья пластмасс под давлением MOLDFLOW (программный продукт компании Autodesk (США)) для создания конечноэлементной модели в ПК SIMULIA Abaqus.

Abaqus/CAE – графическая оболочка для моделирования, управления и мониторинга задач, а также для визуализации результатов расчета в ПК SIMULIA Abaqus.

VCCT – модуль для проектирования и предсказания разрушения и потери несущей способности авиационных конструкций из ламинированных композитных материалов.

В целом, возможности ПК SIMULIA Abaqus включают решение задач, связанных с установившимися во времени и изменяющимися во времени процессами теплопереноса, анализом статического и динамического состояния конструкций, определением собственных частот и форм колебаний, решением задач линейной и нелинейной теорий устойчивости, механики разрушения, моделированием жидких сред, процессов детонации и др.

ПК SIMULIA Abaqus содержит большое количество специальных опций, позволяющих получит решение задач расчета конструкций с учетом разнообразных линейных и нелинейных факторов (пластичность, большие деформации и большие перемещения, гиперупругость, ползучесть, конечные деформации, контактные взаимодействия, термоупругость, анизотропия материала, генетическая нелинейность и др.). По мере развития в программном комплексе был реализован ряд важных дополнительных опций (формирование расчетных схем с использованием подконструкций (суперэлементов), субмоделирование, моделирование случайных колебаний, оптимальное проектирование и др.). Постоянное расширение возможностей способствуют дальнейшему становлению ПК SIMULIA Abaqus как многофункционального аналитического инструмента в различных инженерных приложениях. Следует отметить, что возможности программного комплекса позволяют учитывать целый ряд важных факторов, которым все еще не уделяется должное внимание во многих специализированных пакетах, предназначенных для расчета строительных конструкций, в том числе:

– большие и сверхбольшие размеры моделей и вычислительных задач (до нескольких десятков миллионов узлов, до нескольких десятков тысяч значений собственных частот и форм колебаний, поддержка многопроцессорных расчетов с использованием технологий параллельных вычислений и др.);

– автоматическая генерация сеток конечных элементов на сложных моделях;

– геометрически нелинейный характер деформирования строительных конструкций, зданий и сооружений;

– физически нелинейные свойства строительных материалов (в том числе металлы, сплавы, бетон, железобетон, каменная кладка, кирпичная кладка, дерево, резина, грунтовые основания (включая скальные массивы));

– совместная работа грунтового основания и конструкции, здания, сооружения;

– задание ветровых нагрузок сложного характера со значительной динамической составляющей;

– анализ «внештатных» ситуаций в высоконелинейной динамической постановке с моделированием временных процессов разрушения (в том числе таких как прогрессирующее разрушение, взрыв, цунами);

– решение «нетиповых» задач теплообмена и вентиляции;

– оптимизация проектных параметров нетиповых конструкций.