```
>    tmin:=-8/Pi^2*sum((-1)^n/k^2*tanh(k*z*Pi/2),n=1..N):
>    gamma1[i]:=evalf(tmin/tmax):
>  od:
```

Печать результатов

```
>  x0:=z0,alpha,bet,gamma1:
>  txt:="b/a","alpha","beta","gamma":
>  for j to 4 do
>    printf("%5s|",txt[j]);
>    for i to 9 do
>     printf("%05.3f ",x0[j][i]);
>    od;
>    printf("\n");
>  od;
```

```
  b/a 1.000 1.500 2.000 2.500 3.000 4.000 8.000 10.00 999.0
alpha 0.208 0.231 0.246 0.258 0.267 0.282 0.307 0.312 0.333
 beta 0.141 0.196 0.229 0.249 0.263 0.281 0.307 0.312 0.333
gamma 1.000 0.859 0.795 0.766 0.753 0.745 0.742 0.742 0.742
```

```
>  save beta, "beta.m";
```

5.21. Стесненное кручение тонкостенного стержня открытого профиля

5.21.1. Секториальные характеристики сечения. В первой части программы вводим данные о сечении: размеры `b`, `h`, и толщины `delta1`, `delta2` в сантиметрах, координаты угловых точек и порядок их обхода. Последним (N+1) в списки координат точек входит центр кручения, координаты которого еще не определены и заданы произвольно, в данном случае нулями. Принято, что полюс находится в начале координат. В общем случае это требование излишнее, здесь же это используется для упрощения программы. Особое внимание следует уделить вводу порядка обхода точек, определяемого двойным списком `n`. Начинать обход надо из точки отсчета секториальных координат. Номер этой точки занесен в переменную `n0`. Если сравнивать сечение *открытого* профиля с деревом теории графов [15], то `n0` — корень дерева, а элементы списка `n` — ветви дерева. При вводе есть ограничения:

1. Каждый элемент списка `n` является списком, начинающимся с уже пройденной точки или с `n0`.
2. Каждый участок проходим по одному разу.
3. Проходим все участки.
4. Список должен быть двойным, т.е. если дерево состоит из одной ветки, то его надо искусственно разбить на две.

Программа допускает расчет сечений из элементов разной толщины. Для ускорения ввода сначала все элементы задаются одинаковой толщины δ_2. Это достигается использованием симметричной матрицы `t`, инициированной значением `delta2`. Другие толщины задаются отдельно. В рассматриваемом примере (с. 121) толщина пластины из четырех участков [1,2], [2,3], [3,4] и [4,5] вводится с использованием оператора повторения `$`. Построения эпюры секториальных координат занесено в процедуру `W`, аргументом которой является номер точки полюса. Процедура в программе вызывается дважды. Интегрирование по Верещагину занесено в процедуру `Intgr`, аргументы которой являются списки значений ординат перемножаемых эпюр. Как оператор процедура `Intgr` коммутативна. Если требуется вычислить интеграл не от произведения, а от одной функции, как, например, при вычислении площади, то в качестве второго аргумента берем список из единиц `v1:=Vector(1..N,1)`, т.е. умножаем функцию по правилу Верещагина на эпюру, состоящую из прямоугольников единичной высоты.

Для рисования профиля использован пользовательский оператор `Line` из файла `ris.m`, информация из которого считывается в начале программы. Заметим, что результат существенно зависит от параметра точности счета `Digits`, который лучше не уменьшать от стандарта 10, заданного по умолчанию [1]. Так, очевидно, $\alpha_x = 0$, однако результат получается $0.2824780373\,10^{-8}$, число малое, но отличное от нуля.

Программа позволяет получать результаты и в аналитическом виде, где точность не влияет на результат. Для этого в программе надо закомментировать (знак `#`) оператор вывода изображения `display` или поставить после него двоеточие, так как все изображения в Maple предполагают реальные размеры, выраженные в числах, а не символах. Затем требуется указать тип переменных, например, `assume(h>0,b>0)`. Иначе Maple не будет упрощать выражения, и в промежуточных и конечных результатах будут появляться выражения типа $\sqrt{b^2}$ и никакие операторы упрощения `symplify` тут не помогут. Однако переменные, указанные в операторе `assume` выводятся в листинг по умолчанию со специальным значком $\sim$, что загромождает вывод и затрудняет чтение. Самый удобный (но не единственный) способ борьбы с этой проблемой — использование оператора `interface(showassumed=0)`. При $\delta_1 = = 1.5$ см, $\delta_2 = 1$ см получим аналитические выражения для координаты центра кручения

$$\alpha_y = \frac{h\,(3\,h + 4\,b)}{2\,(19\,b + 3\,h)}$$

[1] Задается параметр точности присваиванием `Digits:=12`, проверяется его значение обычным чтением (точка с запятой после переменной) `Digits`. Для того, чтобы узнать предельно допустимую точность в используемой версии Maple, следует вызвать функцию `kernelopts(maxdigits)`.

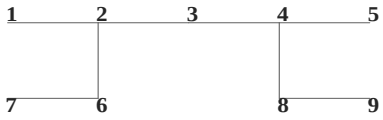
и секториального момента инерции

$$J_\omega = \frac{h^2\,b^2\,(60\,b^2 + 64\,b\,h + 3\,h^2)}{6\,(19\,b + 3\,h)}.$$

Для сокращения текста в программе нет процедур построение эпюр, вместо этого выводятся только списки их ординат. Примеры подпрограмм и процедур построения эпюр см. в программах ref

Программа 27

```
> restart;
> with(plots): with(plottools):
> read "C:\\ris.m";
> # Размеры сечения
> b:=23: h:=21: delta1:=1.5: delta2:=1:
> # Координаты точек сечения
> x:=[-2*b,-b,0,b,2*b,-b,-2*b,b,2*b,0]:
> y:=[0,0,0,0,0,-h,-h,-h,-h,0]:
> N:=nops(x)-1: # Число точек сечения
> n0:=3:# начальная точка
> k0:=3:# полюс (начало координат)
> # Ветви
> n:=[n0,4,5],[4,8,9],[3,2,1],[2,6,7]:
> t:=Matrix(1..N,1..N,delta2,shape=symmetric):
> t[1,2],t[2,3],t[3,4],t[4,5]:=delta1$4:
> k:=nops([n]): # Число ветвей
> Шрифт:=FONT(TIMES,BOLD,8):
> for i to N do
>  Точка[i]:=PLOT(TEXT([x[i]+1,y[i]+1],
>  convert(i,symbol)),Шрифт,COLOR(HUE,0.7)):
> od:
> Ns:=seq(Точка[i],i=1..N):
> Сечение:=seq(seq(Line(n[j,i],n[j,i+1],0.1),
>                  i=1..nops(n[j])-1),j=1..k),Ns:
> display(Сечение,axes=NONE,scaling=CONSTRAINED);
```




```
> W:=proc(m) local i,j,p,q:global w;
> for j to k do
>  w[n0]:=0:
>  for i to nops(n[j])-1 do
>   p:=n[j,i]: q:=n[j,i+1]:
>   w[q]:=(x[q]-x[p])*(y[p]-y[m])-
>                    (x[p]-x[m])*(y[q]-y[p])+w[p];
>  od:
> od:
> end proc:
> W(k0): seq(w[i],i=1..N);
```

$$0,\ 0,\ 0,\ 0,\ 0,\ -483,\ 0,\ 483,\ 0$$

```
> v1:=Vector(1..N,1):
```

Интегрируем по правилу Верещагина

```
> Intgr:=proc(A,B)
> local dst,ds,s,j,i,p,q: global k:
> s:=0:
> for j to k do
>  for i to nops(n[j])-1 do
>   p:=n[j,i]: q:=n[j,i+1]:
>   ds:=sqrt((x[p]-x[q])^2+(y[p]-y[q])^2):
>   dst:=ds*t[p,q]:
>   s:=s+dst/6*(2*A[q]*B[q]+A[q]*B[p]+
>                          A[p]*B[q]+2*A[p]*B[p]):
>  od;
> od; return(s); end:
> F:=Intgr(v1,v1):# Площадь
> Jx:=Intgr(y,y): # Моменты инерции
> Jy:=Intgr(x,x):
```

```
> Sx:=Intgr(v1,y):# Статич.моменты
> Sy:=Intgr(v1,x):
> xc:=Sy/F: yc:=Sx/F:# Центр тяжести
> Jxc:=Jx-F*yc^2: Jyc:=Jy-F*xc^2:
> F_=F, xc_=xc, yc_=yc;Jxc_=Jxc,Jyc_=Jyc;
```

$$F\_ = 226.0000,\ xc\_ = 0.,\ yc\_ = -6.225664$$

$$Jxc\_ = 17700.49115,\ Jyc\_ = 176333.3334$$

```
> x:=evalm(x-xc):  y:=evalm(y-yc):
> Swy:=Intgr(w,x); Swx:=Intgr(w,y);
```

$$Swy := 573965.0000$$

$$Swx := -0.00005$$

```
> alpha_y:=simplify(Swy/Jyc); #Центр кручения
> alpha_x:=-simplify(Swx/Jxc);
```

$$alpha\_y := 3.254999999$$

$$alpha\_x := 0.2824780373\,10^{-8}$$

```
> x[N+1]:=alpha_x-xc; y[N+1]:=alpha_y-yc;
```

$$x_{10} := 0.2824780373\,10^{-8}$$

$$y_{10} := 9.480663716$$

```
> W(N+1):seq(w[i],i=1..N); #Эпюра сект.коорд.
```

$$149.73,\ 74.86,\ 0,\ -74.86,\ -149.73,\ -408.14,\ 149.73, 408.14,\ -149.73$$

```
> D1:=Intgr(w,v1)/F:
> w:=evalm(convert(w,list)-D1):#Эпюра гл. сект.коорд.
> Jw:=Intgr(w,w);
```

$$Jw := 0.497488792\,4\,10^{7}$$

5.21.2. Интегрирование уравнения кручения. Рассмотрим решения задачи на с. 128. Геометрические характеристики J_ω и J_d и секториальные площади в точках K1, K2 (рис. 190, с. 128) могут быть вычислены по программе 27. Все данные и промежуточные приводятся в системе СИ. Дифференциальные уравнения решаются в системе оператором `dsolve`, где в качестве переменной `bc` записаны краевые условия и условия сопряжения. Таким образом, нет необходимости отдельного вычисления констант интегрирования — все вычисления скрыты в операторе `dsolve`. Решение представлено в экспоненциальной форме. Для того, чтобы найденные решения передать искомым функциям $\theta_1(z)$ и $\theta_2(z)$ используется оператор `assign(S)`. После интегрирования $\theta_1(z)$ и $\theta_2(z)$ по z можно построить график угла закручивания по всей длине стержня. Решения на участках склеиваем оператором для получения кусочно-заданных функций `piecewise`.