Applied computer algebra : experience from Cam design

Applied Computer Algebra:

Experience from Cam Design

Arjeh M. Cohen
Fac. Wisk. Inf., Technical University Eindhoven, Postbox 513
5600 MB Eindhoven, The Netherlands

and

André Heck
CAN Expertise Centre, Kruislaan 419, 1098 VA Amsterdam, The Netherlands

ABSTRACT

The construction of a software tool for the design of a cam-follower system is discussed. Although the tool is built with a general purpose computer algebra system, it works as a black-box program. Using this case as an example, it is argued that it is representative of the general scheme: a constructor of mathematical software must — regardless of his own appreciation — pay much attention to the design of the user interface before the software is acceptable for mathematical laymen.

1. Introduction

Although it is generally acknowledged that pure (nowadays often referred to as fundamental) mathematics is one of the most challenging scientific disciplines, its results do not often speak for themselves to the average researcher in industry. In other words, pure mathematical results have not been manifest as extremely useful contributions to society.

One may argue that it is the very nature of pure mathematics not to be applicable. However, the quality of the available software for handling pure mathematics may change the appreciation of what pure mathematics has to offer. In fact it has already done so in certain areas, such as cryptography. There the absence of efficient algorithms for factoring large integers, the availability of well-described implementations of the fastest (publically) known integer factoring algorithms, and the presence of very sophisticated algorithms for checking primality of a number efficiently, have become the key ingredients to widely used cryptographic schemes.

Although this example is taken from a very successful area, it points towards the general, relatively new, phenomenon that, through software, mathematicians have tools to offer *almost* directly to non-mathematically oriented users. In the cryptography example, the key tools come directly from number theory. The mathematician is apt to deliver them as number theoretic software. For such algorithms to become useful tools for cryptography, additional work (such as designing an interface, adjusting the product to the specific situation at hand) has to be done. It is this step, which is not hard, but requires extra work that the mathematician usually neglects or does not care about, that led us to write "*almost* directly" rather than "directly" above. It is an essential one for cryptographic schemes to be used by the mathematical layman.

Similar observations hold for computer algebra. For instance, great progress has been made towards useful tools for exact solving of systems of polynomial equations. But, like the integer factorisation example, the bare product is likely to be cherished by other mathematicians only.

Our experience regarding the general purpose computer algebra packages is that they can be highly useful to those who are scientifically skilled to work with mathematics. Over the last few years, great advances have been made in the development of algorithms for solving mathematical problems other than polynomial system solving, like closed form integration, symmetry analysis, factorisation and treatment of special functions.

In large industries, where there are research departments of considerable size, there will be potential users of such high level tools as the present day computer packages. But in the smaller scale companies, those that cannot afford branches involving mathematically skilled researchers, one may doubt the use of such packages.

There is, however, an indirect way in which general purpose computer algebra packages like Mathematica and Maple may be of use for industrial purposes. The smaller scale industry may have need for certain mathematical "black boxes," which by themselves are produced with the help of a general purpose computer algebra package (compare with the sophisticated methods for primality testing, which the end user need not understand at all). Thus, computer algebra packages become toolkits for manufacturing special-purpose tools of use to non-mathematically oriented users. These dedicated tools will come closer to industrial (end) user's needs and should be manufactured by mathematically oriented researchers.

As small scale industries are usually not capable of handling this step by themselves, and as it usually is not considered to be the academic researcher's task to provide this developmental step, there is a need for special expertise centres that bridge the gap between academia and industry. The CAN Expertise Centre in Amsterdam aims at being such a centre.

Specific examples of what we mean here may be found in the outcome Cohen[2] and Cohen et al[3] of a project called SCAFI (Studies in Computer Algebra For Industry), conducted by the CAN Expertise Centre. To name one, Mathematica is used for symbolically computing the transfer function of a given integrated circuit for loudspeakers, in much the same way, using rewrite rules, that the practitioning circuit designer would go about doing it. This function of Mathematica could be exploited to develop a graphical user interface, in which the designer draws an integrated circuit to be analysed. Then, by the push of a button, the transfer function is computed (at any rate, presently, for circuits of moderate size) and fed into further relevant numerical computations giving the physical characteristics of the loudspeaker for which the circuit was designed.

The remainder of this paper is devoted to the treatment of another case in which the use of computer algebra for the preparation of a tool for the non-mathematical end user (a designer) is prominent.

2. The cam design problem

Crangon BV is a Dutch manufacturer of automatic shrimp peeling machines. An important part of these machines is a V shaped lever that regulates the state of shrimps during the peeling process. The motion of the lever is determined by two rotating cams fixed on a common shaft. The top of the V shaped lever is fixed and each of its legs is in contact with a cam via a circular roller follower. One cam (the "primary cam") is designed to regulate the motion of the lever, whereas the other one (the "secondary cam") is there to ensure that the lever maintains contact. See Figure 1.

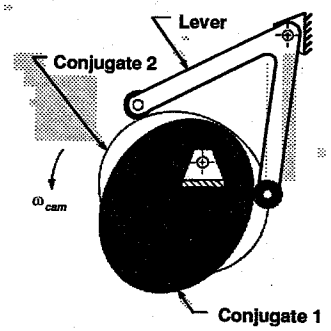

Figure 1: Conjugate Cams on a Common Shaft

By construction, the rim of a cam is built up of line and arc segments, the reason being that a standard milling-machine is capable of automatically milling the cam according to these kinds of segments only.

The problem is, given this primary cam, to compute the rim of the secondary cam. In fact, a recipe is asked for to construct the rim of the secondary cam similarly to the primary one (that is, by milling line and arc segments only).

Thus, the mathematical "black box" that the end user wants delivers data describing the line/arc segment build-up of the rim of a secondary cam, whenever fed the data (of the same type) describing the rim of the primary cam. Here, the end user will be the designer of the peeling machine, a technician who will (interactively) draw the primary cam on his computer screen, and "with the press of a button" call for the blueprint of the secondary cam (thus activating the mathematical black box).

3. The mathematics

The problem is in essence a 2-dimensional one. The primary and secondary cam share an axis perpendicular to the plane we shall be working in, meeting it in what

we shall take to be the origin. Instead of actively rotating the cams, we can (and shall) keep the cams fixed and rotate the top of the lever in the opposite direction. We compute the trajectory of the centre of the circular roller follower that is pressed against the secondary cam. This trajectory is called the *pitch curve* and forms the basis for the computation of the "ideal" secondary cam. At a later stage, we shall discuss a suitable approximation of the resulting pitch curve by a series of line and arc segments, so as to find the required description for the rim of the secondary cam.

The computation of the pitch curve is done by distinction into two cases, according as the type of segment at hand is an arc or a line.

In order to describe the results in more detail, we introduce the following parameters:

- the radii r_1 and r_2 of the first and second circular roller followers;
- the lengths L_1 and L_2 of the first and second leg of the V shaped lever;
- the distance R from the top of the lever to the origin;
- the angle δ between the legs of the lever.

3.1. Arc segments

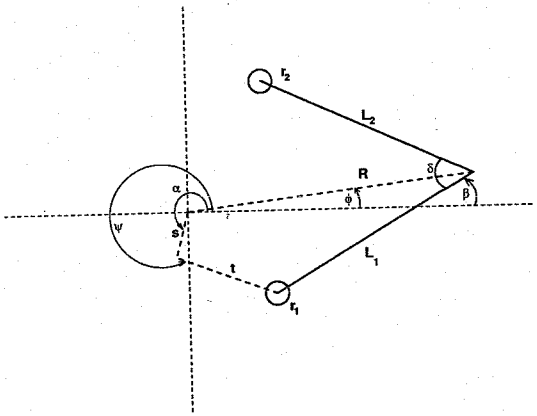

Figure 2: The four-bar linkage associated with arc segment case

We consider the generic case in which the centre of the circle through the arc segment differs from the origin. Then the problem is equivalent to the four-bar linkage problem as drawn in Figure 2. The four bars are:

- the connection, whose length will be denoted by s, from the origin to the centre C_1 of the circle through the arc segment describing the rim of the primary cam;

- the connection from C_1 to the centre of the primary roller follower (the length t of which is equal to the sum of r_1 and the radius of the circle through C_1);
- the first leg of the lever (of length L_1);
- the connection from the top of the lever to the origin (of distance R).

It will be clear that an elementary computation will then give the position of the end of the second leg of the lever, i.e., the coordinates of a point on the pitch curve. Analytic solutions to the four bar problem are known to exist, see Cohen-Heck-Davenport[1] how to find them with the help of a computer algebra system. The bars are of lengths s, t, L_1, and R, respectively, and the angles α, β, ψ, and ϕ are as shown in Figure 2. We give a closed formula for ϕ and β as functions of ψ. In fact, there are two solutions for fixed lengths, but in our arrangement and with clockwise rotation of the cams, only the one given below prevails:

$$\phi = \alpha + 2\arctan\left(\left(\sin(\psi-\alpha) + \left(1 + 2\frac{s\cos(\psi-\alpha)}{t} - \frac{s^2\cos(\psi-\alpha)^2}{R^2} - \frac{s\cos(\psi-\alpha)\left(1+\frac{s^2}{R^2}+\frac{t^2}{R^2}-\frac{L_1^2}{R^2}\right)}{t} + \frac{s^2}{t^2} - \frac{1}{4}\left(1+\frac{s^2}{R^2}+\frac{t^2}{R^2}-\frac{L_1^2}{R^2}\right)\left(1+\frac{s^2}{t^2}+\frac{R^2}{t^2}-\frac{L_1^2}{t^2}\right)\right)^{1/2}\right)\Big/\left(\cos(\psi-\alpha) + \frac{s\cos(\psi-\alpha)}{R} + \frac{s}{t} + \frac{1}{2}\frac{R}{t} + \frac{1}{2}\frac{s^2}{Rt} + \frac{1}{2}\frac{t}{R} - \frac{1}{2}\frac{L_1^2}{Rt}\right)\right)$$

and

$$\beta = \alpha + 2\arctan\left(\left(\sin(\phi-\alpha) + \left(1 - 2\frac{s\cos(\phi-\alpha)}{R} - \frac{s^2\cos(\phi-\alpha)^2}{L_1^2} + \frac{s\cos(\phi-\alpha)\left(1+\frac{s^2}{L_1^2}+\frac{R^2}{L_1^2}-\frac{t^2}{L_1^2}\right)}{R} + \frac{s^2}{R^2} - \frac{1}{4}\left(1+\frac{s^2}{L_1^2}+\frac{R^2}{L_1^2}-\frac{t^2}{L_1^2}\right)\left(1+\frac{s^2}{R^2}+\frac{L_1^2}{R^2}-\frac{t^2}{R^2}\right)\right)^{1/2}\right)\Big/\left(\cos(\phi-\alpha) - \frac{s\cos(\phi-\alpha)}{L_1} - \frac{s}{R} + \frac{1}{2}\frac{L_1}{R} + \frac{1}{2}\frac{s^2}{L_1 R} + \frac{1}{2}\frac{R}{L_1} - \frac{1}{2}\frac{t^2}{L_1 R}\right)\right)$$

It may be worthy of note that if a part of the primary rim is an arc, the curve describing the resulting "ideal" secondary rim is in general not an arc, but a more complex curve.

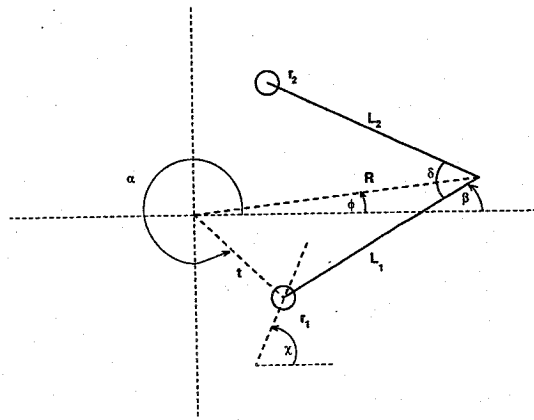

Figure 3: Triangular problem for a line segment

In this case, the solution is analytic again and easier to obtain. It is determined by the cosine rule for a triangle with sides of length t, R, and L_1, respectively. See Figure 3.

$$\begin{aligned} R^2 + t^2 - 2Rt\cos(\phi - \alpha) &= L_1^2 \\ L_1^2 + t^2 + 2L_1 t\cos(\beta - \alpha) &= R^2 \end{aligned}$$

The length t depends on a parameter λ giving the position $(x_0, y_0) + \lambda(\cos\chi, \sin\chi)$ of the centre of the first roller follower.

The resulting formulae for the angles β and ϕ as functions of λ are easily calculated by use of a computer algebra system:

$$\beta = \alpha + 2\arctan\left(\frac{\sqrt{((L_1+t)^2 - R^2)\,(R^2 - (L_1 - t)^2)}}{(R+t)^2 - L_1^2}\right)$$

and

$$\beta = \alpha + 2\arctan\left(\frac{\sqrt{((L_1+t)^2 - R^2)\,(R^2 - (L_1 - t)^2)}}{R^2 - (L_1 - t)^2}\right)$$

where

$$t = \sqrt{(x_0 + \lambda\cos(\chi))^2 + (y_0 + \lambda\sin(\chi))^2}$$

3.3. Approximating the resulting curve

Once the pitch curve is analytically known, the curvature can be computed at each point of the curve. Let $(x(\gamma), y(\gamma))$ denote the position of a point on the pitch curve, where γ ($= \psi$ or λ) parametrises the relevant part of the pitch curve. Let (x', y') and and (x'', y'') denote the velocity and acceleration of the point, respectively. Then, the curvature κ at $(x(\gamma), y(\gamma))$ is given by

$$\kappa = \frac{x'\,y'' - y'\,x''}{(x'^2 + y'^2)^{\frac{3}{2}}}.$$

For each segment of the primary cam we approximate the mean curvature of the corresponding part of the pitch curve by computing the mean value of the curvature in a fixed number of sample points (equally spaced in the parameter space). Next, we compute the centre of the circle that goes through the ends of the part of the pitch curve and has a radius equal to the reciprocal value of the mean curvature. We sum all distances from the sample points to the circle. If they add up to a value smaller than a user specified *tolerance*, we accept the circle part as a good approximation of the corresponding part of the pitch curve. If not, then we split the original line/arc segment of the primary curve into two pieces (by dividing the parameter space into two equal size halves) and treat each piece separately. If the radius of a computed arc segment is larger than a fixed user-defined *threshold*, the arc segment is replaced by a straight line between the end points.

In this way, the pitch curve is approximated by means of arc and line segments. Under the assumption that undercutting is not present, getting from these data to the line and arc segment description of the secondary rim is a matter of shifting and minor further adjustments for the jumps after shift of the segment end points that did coincide before the shift.

3.4. Why computer algebra?

The question may have risen why computer algebra is useful here in the first place? If numeric values would have been obtained for the pitch curve by rotating the V shaped lever over small angles, there would be many segments describing the rim of the secondary cam. In that case the milling machine would not be able to mill the cam as smoothly as is necessary for peeling vulnerable objects like shrimps. Due to symbolic computations, it is possible to give the circles and straight lines directly to the controller of the milling machine and to minimize the number of segments used.

4. The resulting tool

As indicated earlier, the resulting tool, can be fully handled by a cam-follower designer. It works as follows:

The software tool is installed on a PC, where it needs the presence of a DOS-version of Maple (a *Mathematica* version has been implemented as well). It interfaces

with a cam designer's system, called SmartCAM. Thus, the software tool using Maple is actually invoked by pressing a button in a SmartCAM session; it reads in and returns data in a file in SmartCAM's CAMCAD interchange format (in ascii). Such a data file contains lines of the following type:

- in the arc case: ARC $x, y, z, r, \psi_0, \psi_1$,
  where (x, y, z) is the centre of the arc (in solving the problem we ignore the displacement z), r its radius, ψ_0 the angle at starting position, and ψ_1 the angle at the final position of the arc;

- in the line case: LINE x_0, y_0, z, x_1, y_1, z,
  where (x_0, y_0, z) and (x_0, y_0, z) are the starting and end point of the line segment (again we ignore the displacement z).

The cam designer can interactively decide where to write the output data, and which parameter choices for R, L_1, L_2, δ, r_1, r_2 are needed. The computational precision can be set among others with the parameters (*tolerance* and *threshold*, whose interpretations are given in §3.3). Then the black box invokes Maple and outputs data in the same format as it is read in.

Maple's Graphics can be used to picture the primary and secondary cam or to make an animation of the complete system.

This finishes the description of the resulting package. What remains is a description of the Maple code. This will be done in the following section.

5. Maple Code

In this section, we briefly sketch the Maple program to compute the secondary cam. Synoptic descriptions are in **boldface**, further comments are in *italic* font.

5.1. The main program

Read in data regarding the primary cam;
Prepare new data for the centre of the primary roller follower:
  translate segments of primary cam;
  join the newly generated segments;
  determine the data entries (e.g. centres & radii) for the new segments;
Compute the exact pitch curve and its curvature:
for each input segment **do**
  if segment is an arc **then**
    handle arc segment:
      compute 1st and 2nd derivatives of coordinates of pitch curve;
      compute curvature of pitch curve;
      ArcLineApprox: Approximate the pitch curve by arcs and lines;
  else

handle line segment;
fi
od:
Compute segments of secondary cam:
translate segments of centre of secondary roller follower;
join the newly generated segments;
determine the data entries for the new segments;
Write the output data in SmartCAM's CAMCAD interface format.

5.2. Routine to approximate the pitch curve

We finish by describing the Maple routine "ArcLineApprox" alluded to in the main program. It gives an impression of the style of coding. Each inputted segment determines a part of the pitch curve that we shall also call "segment" and that is the argument of the routine call.

```
ArcLineApprox := proc(segment)
  local ...
    x and y are the coordinate functions for points on the pitch curve;
    use them to compute the boundaries of the segment of the pitch curve:
  gamma[0] := op(1,segment);  gamma[1] := op(2,segment);
  X[0] := x( gamma[0] );  Y[0]  := y( gamma[0] );
  X[1] := x( gamma[1] );  Y[1]  := y( gamma[1] );
  mid  := gamma[0]/2 + gamma[1]/2;
    Approximate the mean curvature of the segment of the pitch curve:
  Pieces := 10;  inc := (gamma[1]-gamma[0])/(Pieces+1);
  meanKappa := sum( 'kappa(gamma[0]+j*inc)', 'j'=1..Pieces ) / Pieces;
  signKappa := sign( meanKappa);
  radius := max( 1/abs(meanKappa),
                 sqrt((X[1]-X[0])^2+(Y[1]-Y[0])^2)/2 );
  if radius >= threshold then
    Replace the segment with large radius by a line segment;
  else
    Arc. Find the centre and the boundary values of the output angle:
    [ fsolve( ( X[1] - X[0] + T*(Y[1]-Y[0]) )^2 +
      ( Y[1] - Y[0] - T*(X[1]-X[0]) )^2 - 4*radius^2, T ) ];
    if signKappa>0 then
       T := op( 1, select( x->x>=0, " ) );
    else
       T := op( select( x->x<0, " ) )
    fi;
    Xcentre  := (X[0]+X[1])/2 - T/2*(Y[1]-Y[0]);
    Ycentre  := (Y[0]+Y[1])/2 + T/2*(X[1]-X[0]);
      Compute angles in the range [ 0, 2 Pi ]:
```

```
    PSI[0]  := mod2Pi( arctan( Y[0]-Ycentre, X[0]-Xcentre ) );
    PSI[1] := mod2Pi( arctan( Y[1]-Ycentre, X[1]-Xcentre ) );
      Approximate the mean distance between the arc and the exact pitch curve:
    meanDistance := sum(
      'abs( radius - sqrt( (x(gamma[0]+j*inc)-Xcentre)^2 +
      (y(gamma[0]+j*inc)-Ycentre)^2 ) )', j = 1..Pieces ) / pieces;
    accepted := evalb( meanDistance < tolerance );
    result := [Xcentre, Ycentre, signKappa*radius, PSI[0], PSI[1]]
  fi;
    Check whether approximate curve segment is close enough to the exact pitch
    curve:
  if accepted then
     result
  else
    If not, divide the segment into two parts and repeat the computation:
    mid := (gamma[0] + gamma[1]) / 2;
    ArcLineApprox( [ gamma[0], mid  ] ),
    ArcLineApprox( [ mid, gamma[1] ] )
  fi
end:
```

6. Conclusion

The above worked out case exemplifies that computer algebra software packages are efficient tools for mathematicians to construct software tools for very specialised industrial processes. The mathematics, which otherwise would have been hard to convey to designers in small scale industries, can now be fitted into existing toolkits as an efficient mathematical black box.

7. Acknowledgements

The software has been produced by André Heck. The necessary research was conducted in close collaboration with I. van Woensel and A. Hatvari of Crangon BV.

8. References

1. A.M. Cohen, J.H. Davenport, A.J.P. Heck, *An overview of computer algebra.* In: A.M. Cohen (ed.) *Computer Algebra in Industry* (Wiley, Chichester, 1993) 1–52.
2. Arjeh M. Cohen, ed., *Computer Algebra in Industry*, (Wiley, Chichester, 1993). ISBN 0 471 93829 7
3. Arjeh M. Cohen, Leendert van Gastel, Sjoerd Verduyn Lunel, eds., *Computer Algebra in Industry 2*, (Wiley, Chichester, 1994) to appear.