On the determination of starting points for parametric surface intersections

Karim Abdel-Malek and Harn-Jou Yeh

Two numerical algorithms for computing starting points on the curve of intersection between two parametric surfaces are presented. The problem of determining intersection curves between two surfaces is analytically formulated by parameterizing inequality constraints into equality constraints and augmenting the constraint function. The first method uses an iterative optimization formulation and an iterative conjugate gradient algorithm to minimize a function comprising the vector of coordinates and a weighted constraint term. The second method uses the Moore–Penrose pseudo inverse of the constraint function to determine a starting point. Numerical examples are presented to validate both methods. Both methods require an initial point on one of the surfaces. Numerical examples illustrating the validity of the presented methods are discussed. The local versus the global views of the intersection problem in terms of iterative and recursive subdivision methods are addressed. Difficulties in determining more than one point are also illustrated using examples. The two algorithms are compared by studying their computational complexity. The Moore–Penrose inverse method has showed superior efficiency in the computational complexity, number of iterations needed, and time for conversion to a starting point. It is also shown that the Moore–Penrose inverse converges to a starting point in cases where the iterative optimization method does not. Copyright © 1996 Elsevier Science Ltd

Keywords: surface intersection, parametric surfaces, starting points, Moore–Penrose

In recent years, there have been many studies concerning the determination of intersection curves between surfaces. Bajaj et al. presented a method in which a third-order Taylor approximant is constructed by taking steps of variable lengths and the curve is computed using the general Newton iteration procedure. The method also presents techniques to handle singularities. Montaudoin et al. presented a method where power series are used to approximate plane algebraic curves and surface intersections. General intersection problems have been addressed by Barnhill et al., Barnhill and Kersey, Sederberg, and Farouki.

Lattice evaluation methods were used to determine the intersection curve. Reviews of general intersection methods are numerous. Wilf and Manor presented a method using a modification of Levin’s ruled-surface parametrization scheme, guided by invariant factors classification and furthermore, by factorization of the parametrization polynomials. Markot and Magedson presented a procedural method to parametrize the intersection curve of two surfaces by computing exact points on the true intersection curve. A parallel algorithm using the divide and conquer method was presented by Burger and Schuback. The computational complexity of this algorithm was also analysed. Search techniques were used to refine the interval progressively. Other methods that were applied to the surface-surface intersection problem include a topological and differential-equation method. In this method, the vector field defined as the gradient of the oriented distance function is used to detect critical points in the field such as singularities. Tensorial differential equations are then used to trace intersection segments. Another method that uses unidimensional searches to detect intersection points was presented by Amoura and Uehara. Surface intersection using parallelism were addressed by Chang et al.

A different approach of higher-dimensional formulation including offsets, equal distance surfaces, and variable radius blending surfaces was discussed by Hoffman. A higher-dimensional formulation was also used by Chuang to determine a local and global approximant. Marching methods have been extensively used by researchers in this field. The accuracy of marching methods has been improved by proper control of the step size. Singularities were analysed by Owen and Rockwood, by locally constructing a second-order approximant to each surface. Parametric surface–surface intersection has also been addressed by Houghton et al.

Singularities along the intersection curve have been identified in the work by Lukac who used the quadratic of the curve’s curvature to detect any bifurcation points. Eigenvalues of this system are then studied to determine the form of the matrix, and therefore its behaviour at critical points. Most numerical algorithms require the estimation of a starting point on or close to the solution curve and this is
THE SUBJECT OF THIS PAPER. CUGINI ET AL. 14 INTRODUCED THE CONCEPT OF SHRINKING BOUNDING BOXES THAT COVER THE TOTAL SPACE. PARTS OF THE TWO SURFACES EXISTING IN THE SAME BOUNDING BOX ARE IDENTIFIED TO CALCULATE A STARTING POINT. THE CURVE IS THEN TRACED BY INTRODUCING ONE ADDITIONAL CONSTRAINT AS THE TANGENT TO AN ADVANCING PLANE. THE PROBLEM OF LOOP DETECTION WAS ALSO ADDRESSED USING SIMPLER ALGORITHMS. 15,32,34. ALTHOUGH THESE ALGORITHMS MAY MISS INTERSECTION COMPONENTS, THEY ARE SOFTER, MORE GEOMETRIC, AND IN SOME CASES MORE EFFECTIVE.

MUELLENHEIM 18 PRESENTED AN ITERATIVE METHOD FOR CALCULATING A STARTING POINT THAT IS CLOSE TO A SOLUTION CURVE. THE STARTING POINT HAS BEEN COMPUTED USING LATTICE AND SUBDIVISION METHODS. 15,30,31. ITERATION METHODS WERE RECENTLY USED TO TRACE THE CURVE AND TO DETERMINE TANGENTS AT BIFURCATION POINTS. CURRENTLY, THE MOST WIDELY USED INTERSECTION ALGORITHM IS RECURSIVE SUBDIVISION. 15,16. MAJOR DIFFERENCES BETWEEN ITERATIVE AND RECURSIVE SUBDIVISION TECHNIQUES WILL BE DISCUSSED AFTER INTRODUCING BOTH METHODS.

IN THIS PAPER, TWO METHODS FOR DETERMINING A STARTING POINT ON THE INTERSECTION CURVE ARE PRESENTED. IT WILL BE SHOWN THAT THE MOORE–PENROSE PSEUDO INVERSE PRODUCES MORE EFFICIENT RESULTS THAN AN ITERATIVE OPTIMIZATION FORMULATION. IT WILL ALSO BE SHOWN THAT BOTH METHODS CAN BE USED FOR SPECIAL CASES SUCH AS WHEN THE INTERSECTION LOOP IS VERY SMALL COMPARED WITH THE DIMENSIONS OF BOTH SURFACES.

**CONSTRAINT FORMULATION**

Consider a surface parametrized in terms of independent parameters \( s \) and \( v \) as

\[
\mathbf{x}(u, v) = \mathbf{x}(s, t)
\]

with constrained parameters

\[
\begin{align*}
1 \leq u & \leq 2 \\
1 \leq v & \leq 2 \\
1 \leq s & \leq 2 \\
1 \leq t & \leq 2
\end{align*}
\]

and a second surface parametrized in terms of \( s \) and \( t \) as

\[\mathbf{x}(s, t)\]

with constrained parameters

\[
\begin{align*}
1 \leq s & \leq 2 \\
1 \leq t & \leq 2
\end{align*}
\]

To impose the inequality constraints in numerical form, it is convenient to parametrize the above constraints by introducing new generalized coordinates \( \lambda_i \) such that an inequality constraint of the form

\[
q_i^{min} \leq q_i \leq q_i^{max}
\]

can be parametrized by introducing a new generalized coordinate \( \lambda_i \) as

\[
q_i = a_i + b_i \sin \lambda_i
\]

where \( a_i = (q_i^{max} + q_i^{min})/2 \) and \( b_i = (q_i^{max} - q_i^{min})/2 \) are the midpoints and half-ranges of the inequality constraint.

The intersection problem may be considered as solving the system of seven non-linear equations with eight variables where the constraint function \( \Phi(q) \) is

\[
\Phi(q) = \begin{bmatrix}
x^1(u, v) - x^2(s, t) \\
u - [(u_1 + u_2)/2 + \sin(\lambda_1)(u_2 - u_1)/2] \\
v - [(v_1 + v_2)/2 + \sin(\lambda_2)(v_2 - v_1)/2] \\
s - [(s_1 + s_2)/2 + \sin(\lambda_3)(s_2 - s_1)/2] \\
t - [(t_1 + t_2)/2 + \sin(\lambda_4)(t_2 - t_1)/2]
\end{bmatrix} = 0
\]

where \( q = \{u, v, s, t, \lambda_1, \lambda_2, \lambda_3, \lambda_4\} \) and the inequality constraints of Equations 2, 3, 5, and 6 were parametrized per Equation 8.

If a starting point \( q^0 \) can be found on the intersection curve, Abdel-Malek and Yeh 17 showed that the curve can be traced using continuation methods. To trace along the curve, the tangent vector is needed. The tangent vector is the tangent as a function of the Jacobian of the constraint \( \Phi_q = \partial \Phi / \partial q \), was shown to be uniquely defined by

\[
\Phi_q(q) = 0
\]

the normalization

\[
\| \Phi_q(q) \|_2 = 1
\]

and the determinant

\[
\det \begin{bmatrix}
\Phi_q \\
\Phi_q^T
\end{bmatrix} > 0
\]

The foregoing equations are set to define a unique tangent. The algorithm for calculating this tangent and tangents at bifurcation points was also presented.

**ITERATIVE OPTIMIZATION FORMULATION**

Since the equations represented by the constraint function of Equation 9 are non-linear, a stable computational method is required to obtain a converging estimate of a starting point. It is much more difficult, however, to determine that no solution exists. The problem of concluding that two surfaces do intersect remains unsolved. The objective of the method presented is to minimize the errors in satisfying the constraint function and to find a solution that is close to the initial estimate specified by the user. A technique used in optimization is to successively minimize

\[
\Gamma(q, r) = (q - q^0)^T(q - q^0) + r \Phi^T(q) \Phi(q)
\]

to obtain a starting point with increasing values of the parameter \( r > 0 \) such that the emphasis is concentrated on the constraint equations. The choice of an initial point using this method is extremely delicate. The parameter \( r \) is allowed to grow to a very large number in order to satisfy the constraint equations precisely.

The main idea is to find \( q^*(r) \) to minimize the function \( \Gamma(q, r) \) for a given \( r \) using an optimization algorithm. The parameter \( r \) is then increased and the minimization problem is solved again using \( q^*(r) \) from the previous solution as a starting point. It will be evident that a choice of small \( r \) as a start is necessary to allow it to reach large numbers to achieve convergence. Haug and
Arora\textsuperscript{24} have used this method to solve for an assembled configuration of a mechanism and have suggested increasing \( r \) in discrete steps \( r_i > r_{i-1} \). Convergence of the solution is tested by evaluating the difference between \( q(r_i) \) and \( q(r_{i-1}) \). If the sequence \( q(r_i) \) converges, it is checked by satisfying the constraint. Since there are fewer constraint equations than the dimension of \( q \), there may be many solutions. The minimization solution will produce the first term in Equation 13 that is nearest to the initial point.

To minimize the function of Equation 13, the conjugate gradient minimization algorithm used in non-linear programming will be used\textsuperscript{24,19}. The gradient of the function \( \Gamma(q, r) \) is given by

\[
\frac{\partial \Gamma}{\partial q} = 2(q - q^*)^T + 2r \Phi^T(q) \frac{\partial \Phi(q)}{\partial q}
\]  

(14)

The derivative \( \partial \Gamma / \partial q \) will be abbreviated to \( \Gamma_q \) and similarly the Jacobian \( \partial \Phi / \partial q \) to \( \Phi_q \). The iterative conjugate gradient algorithm was presented by Fletcher and Powell\textsuperscript{19} and is stated in terms of a sequence of computations as follows.

(1) An estimate of an initial point is given by

\[
q^{(0)} = q^*
\]  

(15)

and the matrix of second derivatives of \( \Gamma(q, r) \) is set to identity as

\[
H^{(1)} = H^0 = I
\]  

(16)

where \( I \) is the identity matrix and \( H \) is the matrix of second derivatives of \( \Gamma(q, r) \) such that

\[
H = \frac{\partial^2 \Gamma}{\partial q \partial q^T}
\]  

(17)

(2) The iteration at step \( i \) is computed by the vector \( \eta^i \) such that

\[
\eta^i = -H^{(i+1)} \frac{\partial \Gamma(q^{(i)})}{\partial q}
\]  

(18)

(3) A one-dimensional search algorithm is used to find \( \alpha = \alpha^i \) that minimizes

\[
\min \Gamma(q^{(i)}, \alpha \eta^i)
\]  

(19)

There are many methods that can be used for this step. The one-dimensional Golden Section Search method was used for this purpose. Many one-dimensional algorithms including the Golden Section Search can be found in Arora\textsuperscript{24}.

(4) The value of the generalized coordinates are then computed at \((i + 1)\) as

\[
q^{(i+1)} = q^{(i)} + \alpha \eta^i
\]  

(20)

\[
H^{(i+1)} = H^{(i)} + A^i + C^i
\]  

(21)

where

\[
A^i = \left( \frac{\alpha^i}{\eta^i} \right) \eta^i \eta^T
\]  

(22)

\[
y^i = \Gamma(q^{(i+1)}) - \Gamma(q^{(i)})
\]  

(23)

\[
C^i = -\left( \frac{1}{y^T H^{(i+1)} y} \right) H^{(i+1)} y y^T H^{(i+1)}
\]  

(24)

(5) If \( \frac{\Delta q^{(i+1)}}{\Delta q^{(i)}} = 0 \) or if \( q^{(i+1)} - q^{(i)} \) is sufficiently small, the program is stopped.

Otherwise, the algorithm returns to Step 2 with \( i \rightarrow i + 1 \) until the normal \( \| \Gamma_q(q) \| \) is very small or the algorithm diverges.

In this implementation, it has been our experience that the choice of \( r \) is critical to the speed of convergence of this minimization. The choice of \( r \) in this algorithm has been taken as \( r = 1000 \times 2^{(i-1)} \) where \( i \) is the iteration number.

THE MOORE–PENROSE PSEUDO INVERSE

In this section, the Moore–Penrose pseudo inverse is used to compute a starting point on the intersection curve from an initial point on one of the surfaces. This method has been recently used to compute a starting point for surface–surface intersection\textsuperscript{11}. The method is presented here with its complexity analysis and its comparison with the iterative optimization method presented above. This method is similar to the Newton–Raphson iteration method such that a solution is sought for

\[
\Phi_q \Delta q = -\Phi
\]  

(25)

If the constraint sub-Jacobian matrix is square then Equations 9 and 25 comprise the conventional Newton–Raphson iteration method, with its well-known quadratic convergence properties\textsuperscript{25}.

The constraint equation (Equation 9) has more rows than columns and the constraint sub-Jacobian \( \Phi_q \) has more columns that rows, thus Equation 25 has multiple solutions. One solution to this type of problem is to find the solution \( \Delta q \) with minimum norm \( \| \Phi_q \Delta q \| \), i.e. to solve the minimization problem

\[
\min_{\Delta q} \frac{1}{2} \Delta q^T \Delta q
\]  

(26)

\[
\Phi_q \Delta q = -\Phi
\]  

(27)

Using the Lagrange multiplier approach that appends a multiplier vector times the equations to be satisfied to the function that is to be minimized, define

\[
\Psi \equiv \frac{1}{2} \Delta q^T \Phi \Delta q - \lambda^T (\Phi_q \Delta q + \Phi)
\]  

(28)

As a necessary condition for a solution of the minimization problem, the gradient of the function in Equation 28 is set to zero, yielding

\[
\Delta q^T = \lambda^T \Phi_q
\]  

(29)

Taking the transpose of both sides of Equation 29 yields

\[
\Delta q = \Phi_q^T \lambda
\]  

(30)

Substituting this result into the linearized constraint equations of Equation 27 yields

\[
\Phi_q \Phi_q^T \lambda = -\Phi
\]  

(31)

If the sub-Jacobian \( \Phi_q \) has full row rank, then the coefficient matrix on the left of Equation 31 is nonsingular. In fact, it is positive definite. Therefore,

\[
\lambda = (\Phi_q \Phi_q^T)^{-1} (-\Phi)
\]  

(32)
Substituting this result into Equation 30 yields the result
\[ \Delta q = \Phi_q^{-1}(-\Phi) \] (33)
where the coefficient matrix on the right is the so-called Moore–Penrose pseudoinverse of the sub-Jacobian \( \Phi_q \). Starting with an initial guess \( q^0 \), the update of generalized coordinates are calculated through
\[ \Delta q = \Phi_q^{-1}(-\Phi) \] (34)
where \( \Phi_q \) is the Moore–Penrose pseudoinverse of the Jacobian \( \Phi_q \), defined by Equation 33 as
\[ \Phi_q^{-1} = (\Phi_q^T \Phi_q)^{-1} \Phi_q^T \] (35)
Rather than computing the inverse implied in Equation 35, define
\[ y = (\Phi_q^T \Phi_q)^{-1}(-\Phi) \]
which is equivalent to solving the matrix equation
\[ (\Phi_q^T \Phi_q) y = -\Phi \] (37)
Numerically solving this equation for \( y \) and substituting the result into Equation 19 yields
\[ \Delta q = \Phi_q^{-1} y \] (38)
The set of generalized coordinates can then be computed as \( q^{i+1} = q^i + \Delta q \). Carrying out the Moore–Penrose pseudo inverse iteration using the above method yields a process that has the same quadratic convergence characteristics as the standard Newton–Raphson method.8,9
Using this procedure, the starting point \( q^0 \) on the intersection curve can be found within a few iterations without adding any new constraint equations. Although this method can quickly converge the initial guess into a starting point, only one starting point will be found, and only the corresponding intersection curve can be traced. In addition, specifying an initial point does not guarantee the convergence of this algorithm into a starting point.

COMPLEXITY ANALYSIS

The Moore–Penrose pseudoinverse with Newton–Raphson

The Moore–Penrose method has been recently used in kinematics to compute an initial assembled configuration for mechanisms. It has also been used in continuation methods for the predictor–corrector method by Allgower and Georg.4 Analyses of the computational complexity in terms of number of additions and multiplications required for one iteration for this algorithm are presented in Table 1.

The conjugate gradient minimization algorithm

The complexity analysis presented for this method is only computed for the conjugate gradient minimization algorithm at each optimization iteration \( i \). It is quickly realized that this algorithm is much more complex than the Moore–Penrose method. The computational complexity for this method is presented in Table 2.

In comparing the presented methods with recursive subdivision algorithms, it is emphasized that both the Moore–Penrose pseudo inverse and the iterative

| Table 1 | Computational complexity for the Moore–Penrose pseudoinverse |
|-----------------|-----------------|------------------|
| Operation       | Additions       | Multiplications  |
| Substituting \( q^{i+1} \) into the Jacobian \( \Phi_q \) | \( 8 \times 7 \text{m} \) | 0              |
| Matrix multiplication \( \Phi_q \Phi_q^T \) | \( 8 \times 7 \times 7 \) | \( 8 \times 7 \times 7 \) |
| Solve the linear system \( y \) | \( 7^3 \) | \( 7^3 \) |
| Computing \( \Delta q \) | \( (7 \times 8) \) | \( (7 \times 8) \) |
| Updating \( q^{i+1} \) | 8              | 0              |
| Substituting \( q^{i+1} \) into the constraint functions \( \Phi \) | 8m             | 0              |
| Computing the norm \( \| \Phi(q^{i+1}) \| \) | 7              | 7              |
| Total | \((777 + 64m)\text{n}\) | \(569\text{n} \) |

\( m \) is the number of variables appearing in the Jacobian \( \Phi_q \), and \( n \) is the number of iterations.

| Table 2 | Computational complexity for the iterative optimization method |
|-----------------|-----------------|------------------|
| Operation       | Additions       | Multiplications  |
| Computing \( \partial^1/\partial q \) at \( q^{(i)} \) | 72             | 56              |
| Evaluating \( \eta \) | 64             | 64              |
| Performing the one-dimensional Golden Section Search* (GSS) | \((2 + n_1 + n_2 + n_3) \times 33 \) | \((2 + n_1 + n_2 + n_3) \times 24 \) |
| Updating \( q^{i+1} \) | 8              | 8              |
| Computing \( \partial^1/\partial q \) at \( q^{i+1} \) | 72             | 56              |
| Updating \( y \) | 8              | 0              |
| Evaluating \( \lambda \) | 8              | 137             |
| Evaluating \( C^{(1)} \) | 200            | 264             |
| Computing \( H^{(1)} \) | 128            | 0              |
| Total* | \(560 \times n + \text{GSS}^* \) | \(585 \times n + \text{GSS}^* \) |

* There are three iteration loops in the Golden Section Search. The variables \( n_1, n_2, \) and \( n_3 \) represent the iteration numbers for each loop.
† The total number is multiplied by the number of iterations \( n \) required for conversion.
GSS is the number of additions or subtractions required by the Golden Section Search algorithm.
Table 3 Differences between iterative and recursive methods

<table>
<thead>
<tr>
<th>Finding all solutions</th>
<th>Iteration</th>
<th>Recursion</th>
</tr>
</thead>
<tbody>
<tr>
<td>High performance</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sensitive to erroneous data</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Conjugate methods are iterative methods. A concise comparison between iterative and recursive subdivision algorithms can be found in Dokken et al.15 The general differences are reproduced in Table 3. Since recursive subdivision methods use a 'divide and conquer' algorithm, it is always possible to continue the subdivision until the problem can be classified as simple within a tolerance6. In iterative techniques, however, it is not possible to establish whether more than one solution exists. The Moore-Penrose pseudo inverse, guarantees, however, that if a solution exists it will find it. If multiple solutions exist, the closest solution to the initial guess will be found.

The main difference between iterative and recursive methods is that the former views the intersection problem as a global one and determines whether it has a simple solution. The iterative algorithms view the intersection problem locally. As a result, the recursive algorithms determine all possible solutions, while iterative methods may miss the solution completely.

On the other hand, iterative algorithms exhibit high performance while recursive subdivision algorithms do not. The reason iterative techniques are faster than the recursive techniques is that it is easier to extract local data necessary to march forward in an iteration than the global properties necessary in the recursive subdivision. A disadvantage recursive techniques is their dependence on the results of the subproblem solution. Errors in these solutions will spread to other subproblems.

NUMERICAL EXAMPLES

In this section, numerical examples are presented: (1) to demonstrate the validity of both methods; and (2) to demonstrate their efficiency. Based on our experience using these two methods, the Moore-Penrose pseudo inverse yields better convergence characteristics than those exhibited by the iterative optimization method. In most cases both methods converge. In some cases the Moore-Penrose converges but the iterative optimization method does not. In many cases also, both methods cannot converge. However, there has not been any case where the iterative optimization method converged while the Moore–Penrose did not.

Example 1

Consider the intersection of the following two surfaces. Surface 1 is given by

\[
x'(u,v) = \begin{bmatrix} -3.535 \cos u \cos v + 3.535 \cos u \sin v - 0.606 \cos u + 10.606 \cos u - 5.066 \sin v \\ -3.535 \cos u \sin v - 3.535 \cos u \sin v - 0.606 \sin v + 10.606 \sin v + 3 \end{bmatrix}
\]

with imposed inequality constraints

\[-\frac{\pi}{4} \leq u \leq \frac{3\pi}{4} \text{ and } 0 \leq v \leq 2\pi\]

and surface 2 is given by

\[
x''(s,t) = \begin{bmatrix} 2.500 \cos s \cos t - 2.500 \cos s \sin t - 3.535 \cos s + 0.420 \cos s + 7.500 \cos t \\ 2.500 \cos s \sin t + 3.535 \cos s + 0.420 \cos s + 7.500 \cos t \\ -3.535 \cos s - 3.535 \cos s - 0.606 \cos s + 10.606 \cos s + 20 \end{bmatrix}
\]

with imposed inequality constraints

\[-\frac{\pi}{4} \leq s \leq \frac{3\pi}{4} \text{ and } 0 \leq t \leq 2\pi\]

The two surfaces are shown in Figure 1a. The constraint matrix (Equation 12) can be written as

\[
\Phi(q) = \begin{bmatrix} x'(u,v) - x''(s,t) \\ u - \frac{\pi}{4} - \frac{s}{2} \sin \lambda_1 \\ v - \frac{\pi}{4} - \frac{t}{2} \sin \lambda_2 \\ s - \frac{\pi}{4} - \frac{s}{2} \sin \lambda_3 \\ t - \pi - \frac{t}{2} \sin \lambda_4 \end{bmatrix} = 0
\]

where \( q = [u v s t \lambda_1 \lambda_2 \lambda_3 \lambda_4]^T \). The resulting intersection curves using the a continuation method are shown in Figure 1b.

An initial guess for a point on one of the surfaces (surface 1) is specified as \( q^1 = [\pi/4 \pi/4 \pi/4 0 0 0 0]^T \). Using the iterative optimization formulation, we shall demonstrate the analysis through one iteration. First, a value for \( r \) is chosen (\( r = 1000 \)) and the matrix \( H \) is set to identity as \( H^{(1)} = I \). The vector \( q^{(1)} \) is computed as \( q^{(1)} = -326776.69 - 56066.01 - 326776.69 - 56066.02 0.000 0.000 \). The constant \( \alpha \) is computed using a one-dimensional search algorithm (Golden Section Search15) to be \( \alpha = 0.156 \). The coordinates \( q^{(2)} \) are computed as \( q^{(2)} = [0.276 3.229 0.276 3.054 0 0 0 0]^T \). The vector \( y^{(1)} \) is also computed as \( y^{(1)} = [-285374.98 50785.78 - 285374.98 - 50785.78 1600.485 549.20]^T \). The \( H^{(1)} \) matrix is too small and is not shown. The \( C^{(1)} \) matrix is computed as

\[
\begin{bmatrix} 0.998 & -0.001 & -0.001 & -0.001 & -0.001 & -0.001 & 0.000 & 0.000 \\ 0.000 & 0.998 & -0.001 & -0.001 & -0.001 & -0.001 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.998 & -0.001 & -0.001 & -0.001 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.998 & -0.001 & -0.001 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.998 & -0.001 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.998 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.998 & -0.001 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.998 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.998 \end{bmatrix}
\]

and the \( 8 \times 8 \) \( H^{(2)} \) matrix is computed as

\[
\begin{bmatrix} 0.999 & 1.000 & -0.001 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 1.000 & 1.000 & -0.001 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ -0.001 & 0.001 & 0.999 & 0.999 & 0.999 & 0.999 & 0.999 & 0.999 \\ 0.000 & 0.000 & 0.999 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.999 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.999 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.999 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.999 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \\ 0.000 & 0.000 & 0.999 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix}
\]

The norm computed after the second iteration is \( \| \Gamma_q(q^{(2)}) \| = 59073.6125 \), therefore, the algorithm continues to the third iteration. A total of 14 optimization iterations and 62 iterations in the conjugate gradient minimization algorithm are needed to satisfy a norm of \( \| \Gamma_q(q) \| = 0.53009 \times 10^{-6} \). The final solution converges to the starting point \( q^* = [0.19811 3.10012 0.19811 3.18306 -0.38319 -0.01320 -0.38319 0.01320]^T \).
Starting points for parametric surface intersections: K Abdel-Malek and H-J Yeh

Figure 1  (a) Two parametric surfaces intersecting. (b) Curves due to the intersection of surfaces 1 and 2
To demonstrate the use of the Moore–Penrose generalized inverse with the same initial guess \( \mathbf{q}^0 = [\pi/4 \pi/4 \pi/4 0 0 0 0]^T \) on surface \( \mathbf{1} \), the Jacobian is computed as

\[
\begin{bmatrix}
12.47 & -0.167 \times 10^{-9} & -2.355 & 0 & 0 & 0 & 0 \\
-0.167 \times 10^{-9} & 1.906 & -0.355 & 0.355 & 0.0 & 0 & 0 \\
-2.355 & 0.355 & 3.355 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -10.626 & 7.929 & 0 & 0 \\
0 & 0 & 0 & -7.929 & 7.929 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

The square matrix \( \mathbf{\Phi} \mathbf{\Phi}^T \) is computed as

\[
\begin{bmatrix}
1.067 & 0.006 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.006 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
\end{bmatrix}
\]

Solving the linear system of equations \( \mathbf{\Phi} \mathbf{\Phi}^T \mathbf{y} = -\mathbf{\Phi} \) for \( \mathbf{y} \) yields \( \mathbf{y} = [0.074 \ 0.031 \ 0.256 \ -0.014 \ 0.256 \ 0.014]^T \). Computing \( \Delta \mathbf{q} = -\mathbf{\Phi} \mathbf{y} \) yields \( \Delta \mathbf{q} = [-0.633 \ -0.363 \ -0.363 \ -0.363 \ -0.363 \ -0.363]^T \). The set of coordinates is then updated as \( \mathbf{q}^{(2)} = \mathbf{q}^{(1)} + \Delta \mathbf{q} \), and the norm is calculated as \( \| \mathbf{\Phi} \| = 1.861 \). The procedure is repeated for a total of 4 iterations. At the end of the 4th iteration, the norm is \( \| \mathbf{\Phi} \| = 1.729 \times 10^{-16} \) and the starting point on the curve is computed as \( \mathbf{q}^0 = [0.22111 \ 3.27050 \ 0.22111 \ 3.01268 \ -0.36744 \ 0.04104 \ -0.36744 \ -0.04104]^T \).

**Example 2**

Consider the intersection between the helicoid given by

\[
\mathbf{x}^1(u, v) = [2 + u \sin v \ 3 + u \cos v \ -2 + v/3]^T
\]

and constrained by \(-1 \leq u \leq 1\) and \(-2 \leq v \leq 8\) and the surface \( \mathbf{x}^2(s, t) \) given by

\[
\mathbf{x}^2(s, t) = [2 t^2 \ \sin 3t]^T
\]

constrained by \(0 \leq s \leq 3\) and \(0 \leq t \leq 3\). An initial guess specified in the middle of the interval is given as \( \mathbf{q}^0 = [u \ v \ s \ t]^T = [0 \ 3 \ 1.5 \ 1.5]^T \). After five iterations, a starting point is determined \( \mathbf{q}^5 = [u \ v \ s \ t]^T = [0.2783 \ 3.0841 \ 2.0160 \ 1.6499]^T \). The two surfaces are shown in Figure 2a. The curve of intersection, traced using the continuation method described, is shown on the helicoid in Figure 2b.

**Example 3**

Consider a bicubic Ferguson patch given by

\[
\mathbf{x}^1(u, v) = \mathbf{UFQF}^TV^V
\]

where \( \mathbf{Q} \) has three components \( \mathbf{Q}_x, \mathbf{Q}_y, \) and \( \mathbf{Q}_z \) given by

\[
\mathbf{Q}_x = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 2.5 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ -1 & 2 & 1 & 2 \end{bmatrix}
\]

\[
\mathbf{Q}_y = \begin{bmatrix} 0 & 2 & -1 & 1 \\ 0 & 3 & 0 & 2 \\ 1 & -1 & 2 & -1 \\ 1 & 2 & 1 & 1 \end{bmatrix}
\]

\[
\mathbf{Q}_z = \begin{bmatrix} 0 & -1 & -2 & 1 \\ 0.5 & -0.5 & 1 & -1 \\ 0 & 1 & 2 & -2 \\ 2 & -1 & -2 & 2 \end{bmatrix}
\]

The parameter vectors \( \mathbf{U} \) and \( \mathbf{V} \) are defined by

\[
\mathbf{U} = [1 \ u \ u^2 \ u^3]^T
\]

\[
\mathbf{V} = [1 \ v \ v^2 \ v^3]^T
\]

and the matrix \( \mathbf{F} \) is given by

\[
\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}
\]

The bicubic Ferguson patch is intersected with a bicubic Bezier patch given by

\[
\mathbf{x}^2(s, t) = \mathbf{SBWB}^T \mathbf{T}^T
\]

where \( \mathbf{W} \) has the components \( \mathbf{W}_x, \mathbf{W}_y, \) and \( \mathbf{W}_z \) given by

\[
\mathbf{W}_x = \begin{bmatrix} 1 & 0.5 & 0.6 & 0 \\ 1.5 & 0.7 & 1 & 1 \\ 2 & 1.5 & 2 & 3 \\ 3 & 3.2 & 3.2 & 3.5 \end{bmatrix}
\]

\[
\mathbf{W}_y = \begin{bmatrix} 0 & 0.5 & 0.8 & 2 \\ 0.5 & 0.4 & 1 & 1 \\ -0.5 & 1 & 1.5 & 1.8 \\ 0 & 0.5 & 1.5 & 2 \end{bmatrix}
\]

\[
\mathbf{W}_z = \begin{bmatrix} -1 & 0.5 & -1 & -2 \\ 0 & 0 & 0.5 & 1 \\ 1 & 1 & 1 & 1 \\ 1.5 & 1.4 & 1.6 & 1.5 \end{bmatrix}
\]

and the parametric vectors \( \mathbf{S} \) and \( \mathbf{T} \) are

\[
\mathbf{S} = [1 \ s \ s^2 \ s^3]^T
\]

\[
\mathbf{T} = [1 \ t \ t^2 \ t^3]^T
\]

The matrix \( \mathbf{B} \) is given by

\[
\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}
\]

The two surfaces are shown in Figure 3a. An initial guess
in the middle of the intervals is specified by $q^1 = [0.5 0.5 0.5 0.5]$. After five iterations, the starting point on the curve is computed as $q^2 = [0.3934 0.4638 0.1775 0.4839]^T$. The curve is traced by the continuation algorithm as shown in Figure 3b.

**Example 4**

Consider the intersection of the surface

$$x^1(u, v) = [u \cos v \sin u \sin v \sin u]^T$$

(51)
where $0 \leq u \leq \pi$ and $0 \leq v \leq 2\pi$, and the surface given by

$$\mathbf{x}(s, t) = [3.2 - \sin t \sin 2\sin s, 1.2 + \sin 2t \cos s]^T$$

(52)

where $0 \leq s \leq 2\pi$ and $0 \leq t \leq \pi/2$. The two surfaces are shown in Figure 4a. An initial point in the middle of each interval is specified as $\mathbf{q}^1 = [\pi/2, \pi, \pi/4]^T$. Within five iterations, an initial point is computed as $\mathbf{q}^F = [2.3882, 2.4797, 3.7471, 0.9472]^T$ with a norm of $\|\mathbf{q}^F\| = 7.8 \times 10^{-12}$. The curve of intersection in the $s-t$ space is shown in Figure 4b.

**Example 5**

Consider the intersection of a cylindrical surface with trimmed edges given by

$$\mathbf{x}(u, v) = [6\cos u, 6\sin u, v]^T$$

$u^2 + v^2 = 4$  \hspace{1cm} (53)

where the surface is trimmed by the boundary specified
by $u^2 + v^2 = 4$. The parameters are constrained by $-\pi/2 \leq u \leq \pi/2$ and $-2 \leq v \leq 2$. The second surface is given by

$$x^2(s, t) = [2s - 3 + s \cdot t \cdot \sin(s \cdot t)]^T \quad (54)$$

subject to $2 \leq s \leq 4$ and $0 \leq t \leq 3$. The two surfaces are shown in Figure 5a. An initial guess in the middle of the intervals is specified as $q^0 = [0 \ 0 \ 2 \ 1.5]^T$. After six iterations, a starting point is computed at $q^* = [0.0732 \ -0.6352 \ 2.9712 \ 1.2890]^T$ with a norm of $\| \Phi \| = 6.4 \times 10^{-2}$. The continuation method is used to trace the curve shown on the cylindrical surface in Figure 5b. It is emphasized that the constraint equation due to the trimming curve was handled by the algorithm as an inequality constraint of the form $-\sqrt{4 - v^2} \leq u \leq \sqrt{4 - v^2}$. The algorithm is not affected by trimmed edges.

**Local versus global**

The algorithm presented for determining a starting point is a local method since it can only converge to a starting point that is closest to the initial guess. A starting point can almost always be found if only one curve segment exists. However, in many surface intersection problems it is often the case that many disconnected branches of the curve exist.
Starting points for parametric surface intersections: K Abdel-Malek and H-J Yeh

Figure 5  (a) A trimmed cylindrical patch intersecting with a surface. (b) The curve of intersection on the trimmed surface

Figure 6  Intersection of two surfaces yielding two disconnected branches
exist. For this case, starting points may be found, each on a branch, if different initial guesses are specified. Therefore, a global solution for the starting point cannot be found.

**Example 6**

For example, consider the intersection of a surface parametrized by \( x^1(u, v) = [u \ a \ (u-\gamma_1)[u^2-v^2]-\gamma_2v^2]^T \) subject to \(-0.5 \leq u \leq 1.5\) and \(-1 \leq v \leq 1\), where \( \gamma_1 = 0.5 \) and \( \gamma_2 = 0.6 \), and a second surface given by the plane \( x^2(s, t) = [s \ t \ 0]^T \) as shown in Figure 6. Two different initial guesses are necessary to yield two starting points for the continuation algorithm. It is emphasized that an initial point close to one curve segment yields a starting point on that segment.

**Example 7**

A similar situation arises for the case of the two surfaces shown in Figure 7a represented by

\[
x^1(u, v) = \begin{bmatrix} 0.5 \cos(0.5\pi u) \cos(\pi v) \\ 0.5 \cos(0.5\pi u) \cos(2\pi v) \\ 0.5 \sin(\pi v) \end{bmatrix}^T
\]

subject to \(0 \leq u \leq 1\) and \(0.5 \leq v \leq 0.5\), with the surface defined by

\[
x^2(s, t) = \begin{bmatrix} 0.5 \cos(\pi(1 + \frac{1}{2}s)) \cos(\pi t) + a \\ 0.5 \sin(\pi(1 + \frac{1}{2}s)) \cos(\pi t) + a \\ 0.5 \sin(\pi t) \end{bmatrix}
\]

where \(a = 0.35355\) subject to \(0 \leq s \leq 1\) and \(0.5 \leq t \leq 0.5\).

If the variable \(a\) in Equation 56 is changed to \(a = 0.0702101\), it results in the case of no intersection between the two surfaces as shown in Figure 7b. In fact, this case was studied by Mullenheim\(^5\) where the algorithm recognized that there was no intersection at the sixth iteration. Using the Moore–Penrose method, the norm of the constraint function \(\| \Phi \|\) starts increasing indicating a divergence.

**Tracing difficulties**

In some cases, although there exists one curve of intersection between the two surfaces, and although a starting point can be readily determined, it is not possible to continue tracing the curve. In this case, the curve is discontinuous at a point that exhibits a higher order of bifurcation and two tangents must be determined. In this case, however, only one tangent exists. The tracing method stops at the bifurcation point. Another initial guess is specified to determine a starting point on the other segment. Recognizing that there exists another segment cannot be established using the Moore–Penrose inverse.

**Example 8**

To illustrate, consider the intersection of the surface given by

\[
x^1(u, v) = [u \ v \ u^2 + (u - 1)v^2]^T
\]

subject to \(-1 \leq u \leq 1\) and \(-1 \leq v \leq 1\), with the plane given by \(x^2(s, t) = [s \ t \ 0]^T\). This case was studied by Mullenheim\(^5\). The two surfaces are shown in Figure 8a. The intersection curve has two segments as shown on the \(u-v\) space in Figure 8b. A starting point on either segment of the curve can readily be determined. The numerical continuation algorithm can only trace the segment until it reaches the bifurcation point. At this point, only one tangent exists, therefore the continuation method cannot continue along the second segment of the curve. A second initial guess is specified and a starting point on the second segment is computed. The second segment is then numerically traced.
Algorithm efficiency

In order to measure the efficiency exhibited by each algorithm, the time required for each method to converge was monitored. Using an HP 735-125 machine, the following results were obtained by varying the coordinates of the initial point on one of the surfaces. The point coordinates are changed such that the distance of that point from the curve is increased. The results are presented in Table 4.

It is evident that the time required for the iterative optimization formulation is large compared with that required by the Moore–Penrose pseudo inverse method. It is also evident that, as the initial point is selected further away from the curve, it takes longer for both methods to converge. Much longer is needed, however, for the iterative optimization method since for each iteration it performs the algorithm has to perform the one-dimensional searching routine, and for each iteration the value of $r$ is increased. The iterative optimization failed to converge in three cases while the Moore–Penrose always converged.

### CONCLUSIONS

The two numerical algorithms presented in this paper and illustrated using numerical examples of parametric surface intersections with inequality constraints imposed on their parameters demonstrate the feasibility of a general numerical method for determining starting points on the curve of intersection. Both methods have been shown to converge to a starting point that can be used for tracing curves in an iterative algorithm.

It has been shown that both methods are relatively efficient by comparing their complexity analyses. It has also been shown that the Moore–Penrose method demonstrates superior efficiency to that demonstrated by the iterative optimization formulation in the number of iterations and time required to converge to a starting point.

It is also our observation that the iterative optimization technique converges for initial points that are close to the curve. If this method converges, it usually requires at least 60 iterations. It is also highly dependent on the weight factor $r$. The choice of a small $r$ is recommended in order to span a large range of values for $r$, to a value

<table>
<thead>
<tr>
<th>Initial point position $(u, v)$</th>
<th>Time (s) for iterative optimization</th>
<th>Time (s) for Moore–Penrose inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2, $\pi$</td>
<td>203.6</td>
<td>2.2 (second iteration)</td>
</tr>
<tr>
<td>0.3, $\pi$</td>
<td>304.0 (18 optimizations, 82 loops)</td>
<td>2.2 (third iteration)</td>
</tr>
<tr>
<td>0.4, $\pi$</td>
<td>Diverged</td>
<td>2.1 (third iteration)</td>
</tr>
<tr>
<td>0.5, $\pi$</td>
<td>445.9 (25 optimizations)</td>
<td>2.2 (third iteration)</td>
</tr>
<tr>
<td>0.6, $\pi$</td>
<td>327</td>
<td>2.3 (third iteration)</td>
</tr>
<tr>
<td>0.7, $\pi$</td>
<td>483.6 (29 optimizations)</td>
<td>2.4 (fourth iteration)</td>
</tr>
<tr>
<td>0.8, $\pi$</td>
<td>Diverged</td>
<td>2.4 (fourth iteration)</td>
</tr>
<tr>
<td>0.9, $\pi$</td>
<td>Diverged</td>
<td>3.5 (fourth iteration)</td>
</tr>
</tbody>
</table>
Starting points for parametric surface intersections: K Abdel-Malek and H-J Yeh

that causes the algorithm to converge (usually a large value). Starting with a higher value for r may lead to a faster convergence but may risk diverging. Both methods may detect special cases such as when the two surfaces are tangent at a point.

Iterative type algorithms such those presented in this paper are local. Starting points close to the initial guess can be found. Difficulties encountered due to the existence of multiple disconnected branches or when the curve has two segments connected at a higher order bifurcation point were addressed.

ACKNOWLEDGEMENT

This research was supported by the NSF-Army Industry/University Cooperative Research Center I/UCRC at the Center for Computer Aided Design.

REFERENCES

Starting points for parametric surface intersections: K Abdel-Malek and H-J Yeh

47. Varady, T., Surface-surface intersections for double-quadratic parametric patches in a solid modeler. Proceedings of the UK

Hungarian Seminar on Computational Geometry for CAD/CAM.
Cambridge University, Cambridge, UK, 1983.

Karim Abdel-Malek is assistant professor of Mechanical Engineering at the University of Iowa. He received his BS degree from the University of Jordan and his MS and PhD degrees from the University of Pennsylvania, in mechanical engineering. Dr. Abdel-Malek consulted for the manufacturing industry for a number of years before joining the faculty at the University of Iowa. His research interests include CAD/CAM systems, robotics, dynamics, and machine design.

Harn-Jou Yeh is a PhD candidate in the Department of Mechanical Engineering at the University of Iowa. Mr. Yeh received his BS degree from the National Taiwan University and his MS degree from the University of Iowa. He has worked for a number of years for the Center of Computer Aided Design. Mr. Yeh's research interests are in kinematics, computer-aided geometric design, and robotics.