A Maple/Matlab toolbox for Computer Aided Geometric Design

Laureano González-Vega[‡], Ioana Necula[*] y David Sevilla
Departamento de Matemáticas, Estadística y Computación
Universidad de Cantabria, Spain
{gvega,ioana,sevillad}@matesco.unican.es

Abstract

In this report we survey an ongoing work devoted to show how the already widely used Scientific Computing Systems (in our case Maple and Matlab) integrating symbolic and numeric capabilities can be used to develop a Problem Solving Environment very useful to solve problems into a CAD/CAM framework. Second section shows how algebraic techniques and Scientific Computing Systems can be very useful in CAGD. The remaining sections of this paper are Maple spreadsheets (with some calls to Matlab to solve some huge linear systems of equations) where some concrete problems in Computer Aided Geometric Design are solved.

1 Introducción

Este articulo se describen un trabajo en curso dedicado a mostrar como el uso de sistemas de Calculo Cientifico que integran facilidades numericas y simbolicas como, en nuestro caso, Maple y Matlab pueden ser muy utiles a la hora de resolver problemas reales en entornos CAD/CAM.

La segunda seccion muestra como las tecnicas algebraicas y los sistemas de claculo cientifico como Maple y matlab pueden ser muy utiles en CAGD. Las secciones restantes de este informe son sesiones de Maple (Maple spreadsheets), con llamadas internas a Matlab para resolver algunos sistemas de ecuaciones lineales de gran tamao, resolviendo algunos problemas tipo en Diseo Geometrico asistido por ordenador (CAGD en la terminologia al uso). Mas precisamente la primera seccion se dedica a el calculo, topologicamente exacto, del seccionado de una superficie de revolucion. Las secciones 4, 5 y 6 muestran como trabajar en Maple con curvas y superficies B-spline.

2 Problemas algebraicos en CAGD

La utilidad de los sistemas de CAD/CAM como un medio para incrementar la eficiencia en los procesos de simulación y diseño dentro del sector productivo es en la actualidad irrebatible. Ventajas como la reducción en tiempo de producción, mejora en la calidad del producto final y reducción de costes al disminuir el tiempo de implementación de cambios en el proceso de diseño son frecuentemente citadas como los mayores beneficios que produce la introducción de los sistemas de CAD/CAM en un entorno industrial.

En este trabajo se produce la convergencia de tres lineas de investigacion:

- El estudio y potenciación de las herramientas gráficas que proporcionan los sistemas actuales de software matematico de proposito general (`Mathematica`, `Matlab`, `Maple` y `Axiom`) junto con el desarrollo, en estos sistemas, de módulos de simulación específicos para modelado geométrico y visualización.

[*]Partially supported by DGESIC PB 98-0713-C02-02 (Ministerio de Educación y Cultura)
[†]Partially supported by the FEDER Project 1FD97-0409 (Ministerio de Educación y Cultura)

- La adecuación, desarrollo e integración de las técnicas de manipulación de los conjuntos solución de sistemas de ecuaciones algebraicas desarrolladas en el marco del proyecto FRISCO (ESPRIT/LTR 21024: Unión Europea) a la resolución eficiente de problemas en la manipulación de curvas y superficies parametricas.

- La resolución de un conjunto muy concreto de problemas, no resueltos satisfactoriamente en la actualidad, para el entorno CAD/CAM CSIS que utiliza la empresa CANDEMAT en los procesos de creación de matriceria de chapas para la carroceria de vehículos. Esta resolución se aborda mediante la inclusión de técnicas desarrolladas sobre los módulos de simulación mencionados anteriormente y/o a través de la incorporación de técnicas o software ya contemplados en el marco del proyecto FRISCO.

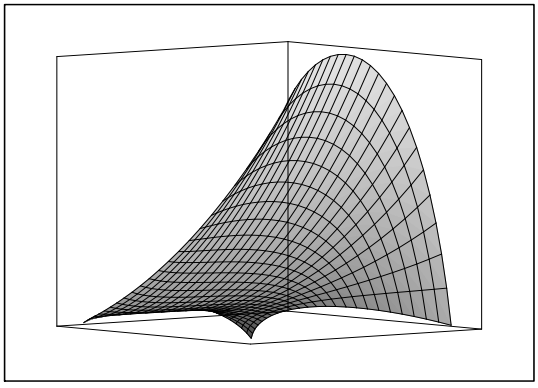
Tradicionalmente los sistemas de CAGD se han desarrollado en lenguajes estándar con buenas características para el cálculo científico (C, C++, Fortran, Basic, Pascal,...). Así como se está produciendo una evolución muy rápida en la tecnología de hardware, la evolución del software sigue un camino más lento, aunque también continuo. Este trabajo se centra en concreto a los paquetes integrados de cálculo simbólico y numérico, con una elevada capacidad gráfica. Esta modalidad de software está alcanzando gran difusión y relevancia en los últimos años. Junto a sus instrucciones generales básicas, estos sistemas ofrecen módulos adicionales que incorporan aplicaciones más específicas. Tal sucede con los "Packages" en Mathematica y las "ToolBoxes" en Matlab.

En Mathematica, hay un "Package" denominado Graphics 'Spline' con una capacidad limitada para generar como entidades básicas curvas Bezier y splines cúbicos. Las potentes capacidades de gráficos en paramétricas, operadores, etc, pueden permitir aplicar estas primitivas para modelados interesantes, pero con alcance limitado. En MATLAB hay una función básica para generar splines cúbicos, y desde 1990 la Toolbox 'Spline' en que se presentan las funciones B-splines para curvas y superficies, siguiendo la obra clásica de Carl de Boor.

Por otra parte en los últimos años, se ha realizado un gran esfuerzo investigador, dentro de la comunidad cientifica internacional, en la busqueda de algoritmos/métodos eficientes (i.e. rápidos) y lo más precisos posibles (validez garantizada de las soluciones obtenidas) para la resolución de ecuaciones algebraicas (o polinomiales) y la manipulacion de los conjuntos de sus soluciones. El significado preciso de la frase anterior puede observarse mas cláramente en el siguiente ejemplo: considera la ecuación paramétrica de la superficie bicúbica $\mathcal{S}$

$$
\begin{aligned}
x &= 3t(t-1)^2 + (s-1)^3 + 3s \\
y &= 3s(s-1)^2 + t^3 + 3t \\
z &= -3s(s^2-5s+5)t^3 - 3(s^3+6s^2-9s+1)t^2 + t(6s^3+9s^2-18s+3) - 3s(s-1)
\end{aligned}
$$

para valores de s y t en el intervalo $[0,1]$ y cuyo aspecto aparece a continuación:



Un primer problema que puede aparecer en esta situación es la determinación de los puntos intersección de$\mathcal{S}$ con la recta $x = u, y = u, z = u$ (si es que tal intersección existe). En definitiva esto no es más que la

resolución del sistema de ecuaciones (no lineal):

$$
\begin{aligned}
u &= 3t(t-1)^2 + (s-1)^3 + 3s \\
u &= 3s(s-1)^2 + t^3 + 3t \\
u &= -3s(s^2 - 5s + 5)t^3 - 3(s^3 + 6s^2 - 9s + 1)t^2 + t(6s^3 + 9s^2 - 18s + 3) - 3s(s-1)
\end{aligned}
$$

En este caso particular se obtiene que sólo existe un punto de corte con coordenadas

$$(0.5561, 0.5561, 0.5561)$$

alcanzado para $s = 0.2748$ y $t = 0.0408$. Otra forma más inteligente de resolver este problema se apoya en la determinación de la ecuación implicita de $\mathcal{S}$: esto es, la ecuación $H(x, y, z)$ en las variables x, y y z que satisfacen los puntos de $\mathcal{S}$. Si tal ecuación ha sido calculada el problema anterior se reduce a resolver la ecuación $H(u, u, u) = 0$ y análogamente todo problema de intersección de $\mathcal{S}$ con cualquier recta, o en general, cualquier curva ("spline", por ejemplo) se reduce a resolver una ecuación en una variable. En este caso particular, la ecuacion implicita de $\mathcal{S}$ posee la siguiente estructura

$$H(x, y, z) = z^9 + \sum_{i=1}^{9} r_i(x, y) z^{9-i}$$

$$
\begin{aligned}
\mathbf{r}_1(x,y) &= -\tfrac{233469x}{2048} + \tfrac{188595y}{2048} - \tfrac{112832595}{262144} - \tfrac{81x^2}{64} + \tfrac{135xy}{32} - \tfrac{81y^2}{64} \\
\mathbf{r}_2(x,y) &= -\tfrac{20972672709381x}{536870912} + \tfrac{17975329363179y}{536870912} - \tfrac{729y^4}{8192} - \tfrac{729x^4}{8192} + \tfrac{1215x^3y}{2048} - \tfrac{4779x^2y^2}{4096} \\
&\quad + \tfrac{1215xy^3}{2048} - \tfrac{4105971x^3}{65536} + \tfrac{3129597y^3}{65536} + \tfrac{14456151x^2y}{65536} - \tfrac{13181049xy^2}{65536} - \tfrac{54187594407x^2}{16777216} \\
&\quad + \tfrac{48101467761xy}{8388608} - \tfrac{38812918311y^2}{16777216} - \tfrac{22656991982391171}{137438953472} - \tfrac{1}{2}\left(\tfrac{233469x}{2048} - \tfrac{188595y}{2048}\right. \\
&\quad \left. + \tfrac{112832595}{262144} + \tfrac{81x^2}{64} - \tfrac{135xy}{32} + \tfrac{81y^2}{64}\right)\left(-\tfrac{233469x}{2048} + \tfrac{188595y}{2048} - \tfrac{112832595}{262144} - \tfrac{81x^2}{64}\right. \\
&\quad \left. + \tfrac{135xy}{32} - \tfrac{81y^2}{64}\right) \\
\mathbf{r}_3(x,y) &= \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned}
$$

y la sustitución $x = u, y = u, z = u$ proporciona una ecuación en u de grado 18 muy sencilla de resolver:

$$5159780352u^{18} - 609499054080u^{17} + \ldots\ldots\ldots + 37079122734922422562595663 13 = 0$$

El problema de esta filosofía de trabajo se encuentra en que a pesar de que existen métodos para calcular $H(x, y, z)$, estos son muy ineficientes y, por ello, dificilmente integrables en los paquetes de software CAD/CAM donde las soluciones a cualquier demanda del usuario han de producirse en tiempo real. Sin embargo, lo que si es factible es la incorporacion de una base de datos con las implicitaciones (ya calculadas y preporocesadas) de las superficies parametricas de uso mas frecuente: por ejemplo la ecuacion implicita de la superficie parametrica

$$
\begin{aligned}
x = &\quad x_{00}\frac{t_2 - t}{t_2 - t_1} + x_{01}\frac{t - t_1}{t_2 - t_1} \\
y = &\quad y_{00}\frac{s_2 - s}{s_2 - s_1} + y_{11}\frac{s - s_1}{s_2 - s_1} \\
z = &\quad \left(z_{00}\frac{s_2 - s}{s_2 - s_1} + z_{10}\frac{s - s_1}{s_2 - s_1}\right)\frac{t_2 - t}{t_2 - t_1} + \left(z_{10}\frac{s_2 - s}{s_2 - s_1} + z_{11}\frac{s - s_1}{s_2 - s_1}\right)\frac{t - t_1}{t_2 - t_1}
\end{aligned}
$$

es

$$
\begin{aligned}
&(z_{00} - 2z_{10} + z_{11})xy + (y_{00}z_{10} + y_{11}z_{10} - y_{00}z_{11} - y_{11}z_{00})x \\
&\quad + (x_{00}z_{10} + x_{01}z_{10} - x_{00}z_{11} - x_{01}z_{00})y + (x_{00}y_{11} - x_{00}y_{00} + x_{01}y_{00} - x_{01}y_{11})z \\
&\quad + x_{01}y_{11}z_{00} - x_{00}y_{11}z_{10} + x_{00}y_{00}z_{11} - x_{01}y_{00}z_{10}
\end{aligned}
$$

independientemente de los valores de los parametros x_{ij}, y_{ij} y z_{ij}. Asimismo es posible determinar las expresiones algebraicas que describen la dependencia de s y t en funcion de x, y y z.

Este problema, y todo tipo de problemas cuya formulación y resolución conlleve la manipulación de sistemas de ecuaciones no–lineales (en las que pueden intervenir o no parámetros) son jústamente a los que se ha dedicado nuestro grupo de trabajo en el marco del proyecto europeo FRISCO (A **Fr**amework for **I**ntegrated **S**ymbolic/Numeric **Co**mputation, Marzo 1996 a Marzo 1999). Los problemas tratados en las siguientes secciones son un ejemplo de la utilización de las técnicas desarrolladas en dicho proyecto a la resolución de problemas en CAGD.

3 Seccionado de superficies de revolución

En esta seccion se utilizan una serie de tecnicas algebraicas (subresultantes, principalmente) para determinar de forma exacta el seccionado de una superficie de revolución.

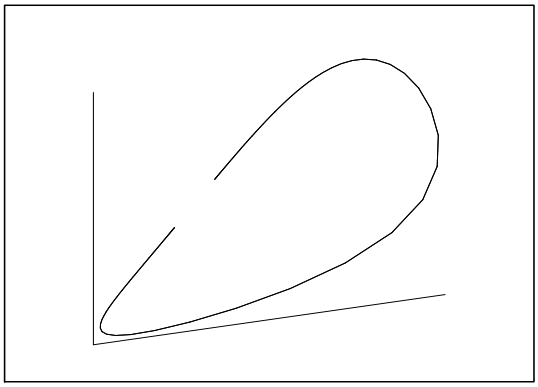
La curva C en el plano $Y = 0$ y definida por la parametrización $x = C_1(t)$, $z = C_2(t)$ va a ser rotada respecto del eje OZ.

```
> plots[setoptions3d](scaling=CONSTRAINED,axes=FRAMED);
> plots[setoptions](scaling=CONSTRAINED,axes=FRAMED);
> C1:=(2*t-1)/(1+t**2);C2:=0;C3:=(1-t+t**2)/(2+t+t**2);
> plot3d([C1,C2,C3],t=-15..15,s=-1..1,orientation=[60,75],grid=[175,2]);
```

$$C1 := \frac{2\,t-1}{1+t^2}$$

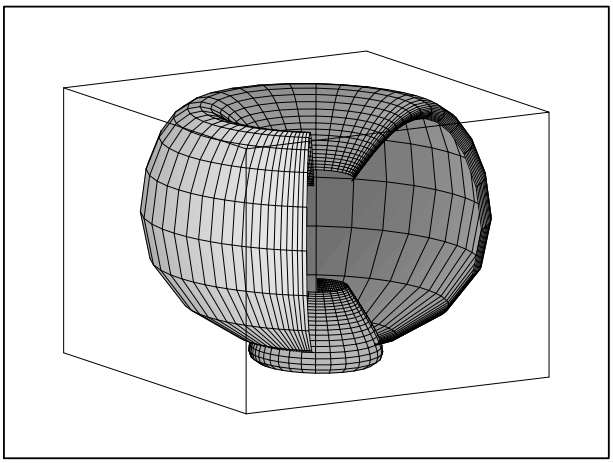
$$C2 := 0$$

$$C3 := \frac{1-t+t^2}{2+t+t^2}$$



A continuación se muestra la gráfica de esta superficie de revolución generada por la curva C.

```
> T1:=C1*(2*s)/(1+s**2)-C2*(1-s**2)/(1+s**2):
> T2:=C2*(2*s)/(1+s**2)+C1*(1-s**2)/(1+s**2):
> T3:=C3:
> P1:=(a,b)->plot3d([T1,T2,T3],s=a..b,t=-6..6,grid=[50,50],axes=BOXED,s
> caling=UNCONSTRAINED,projection=1,tickmarks=[0,0,0],orientation=[59,75
> ]):
> P1(-3,3);
```



Para estudiar las secciones de esta superficie se calcula en primer lugar su ecuación implícita.

```
> read "ImplicitSuperfRevol.txt";
> toro:=ImplicitSuperfRevol(C1,C2,C3,x,y,z,t);
```

$$\begin{aligned} toro := {} & 9 + 8\,z\,x^2\,y^2 + 262\,z^2 - 308\,z^3 - 84\,z + 8\,z^4\,x^2\,y^2 + 121\,z^4 + 16\,z^2\,x^2\,y^2 + 20\,z\,x^2 \\ & + 20\,z\,y^2 - 3\,x^2 - 3\,y^2 + x^4 + y^4 + 16\,z^3\,x^2\,y^2 + 19\,z^4\,x^2 + 19\,z^4\,y^2 + 2\,x^2\,y^2 + 4\,z^4\,x^4 \\ & + 4\,z^4\,y^4 + 8\,z^3\,x^4 + 8\,z^3\,y^4 - 72\,z^3\,x^2 - 72\,z^3\,y^2 + 8\,z^2\,x^4 + 8\,z^2\,y^4 - 28\,z^2\,x^2 \\ & - 28\,z^2\,y^2 + 4\,z\,x^4 + 4\,z\,y^4 \end{aligned}$$

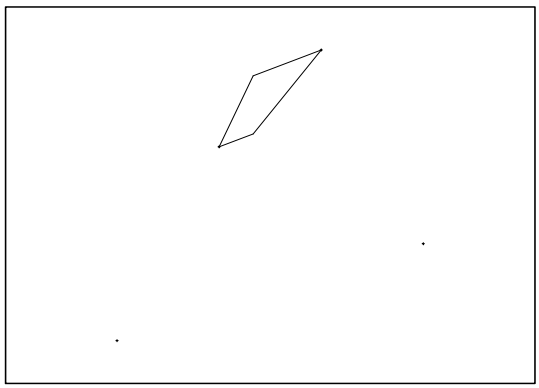
Se comienza calculando la sección de esta superficie por el plano $y = 1$.

```
> Seccion1:=subs(y=1,toro);
```

$$\begin{aligned} Seccion1 := {} & 7 + 242\,z^2 - 372\,z^3 - 60\,z + 144\,z^4 + 28\,z\,x^2 - x^2 + x^4 + 27\,z^4\,x^2 + 4\,z^4\,x^4 \\ & + 8\,z^3\,x^4 - 56\,z^3\,x^2 + 8\,z^2\,x^4 - 12\,z^2\,x^2 + 4\,z\,x^4 \end{aligned}$$

Se determina inicialmente la topologıa de esta sección (notar que tiene dos puntos aislados):

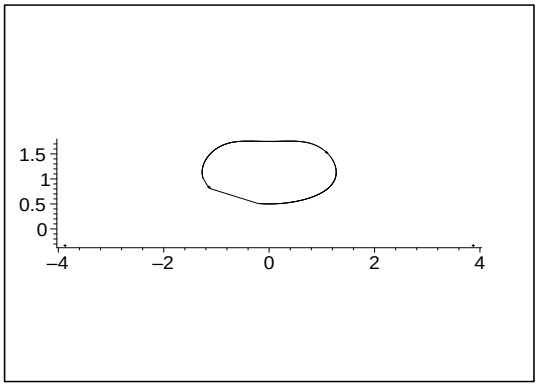
```
> read "prGrafos.txt";
> principal(Seccion1,x,z,20,'black');
```

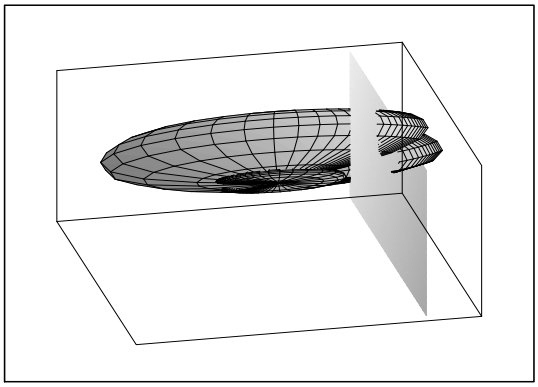


para a continuación proceder a mostrar a su dibujo exacto:

```
> read "prDibujos.txt";principal(Seccion1,x,z,20,500,10,'black');
> P2:=y->plot3d([s,y,t],s=-4..4,t=-2..2,style=PATCHNOGRID):
```

```
> plots[display](P1(-4,4),P2(1),grid=[50,50],axes=BOXED,scaling=UNCONST
> RAINED,projection=1,tickmarks=[0,0,0],orientation=[13,130]);
```





La siguiente sección a considerar viene determinada por el plano $y = 1/2$:

```
> Seccion2:=subs(y=1/2,toro);
```

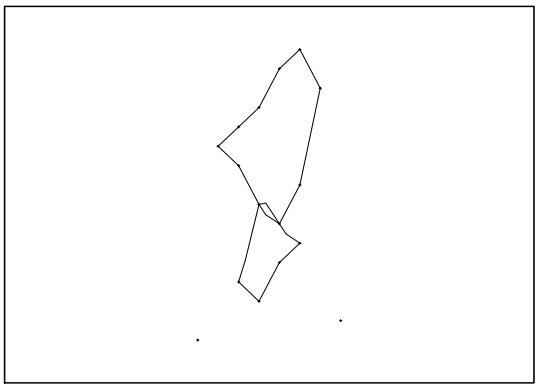
Se determina inicialmente la topología de esta sección:

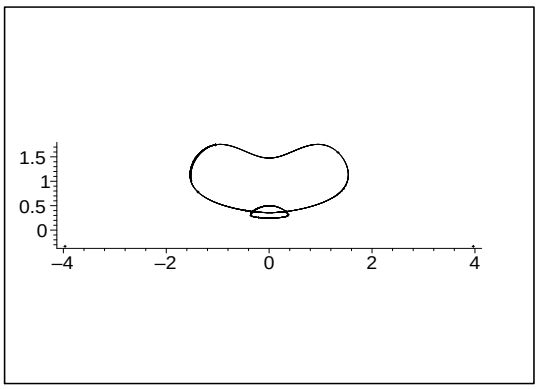
```
> read "prGrafos.txt";
> principal(Seccion2,x,z,20,'black');
```

para a continuacion proceder a mostrar a su dibujo exacto:

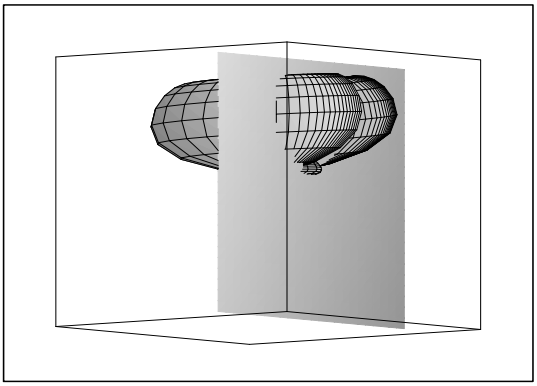
```
> read "prDibujos.txt";
> principal(Seccion2,x,z,20,1000,10,'black');
```

$$Seccion2 := \frac{511}{2}\,z^2 - \frac{651}{2}\,z^3 + \frac{133}{16} - \frac{315}{4}\,z + 126\,z^4 + 22\,z\,x^2 - \frac{5}{2}\,x^2 + x^4 + 21\,z^4\,x^2 + 4\,z^4\,x^4 + 8\,z^3\,x^4 - 68\,z^3\,x^2 + 8\,z^2\,x^4 - 24\,z^2\,x^2 + 4\,z\,x^4$$






```
> plots[display](P1(-4,4),P2(1/2),grid=[50,50],axes=BOXED,scaling=UNCON
> STRAINED,projection=1,tickmarks=[0,0,0],orientation=[40,95]);
```



4 Generacion aleatoria y dibujo de curvas y superficies B-spline (racionales y polinomiales)

Se generan (simbólicamente) una curva B-spline polinomial y otra racional (generadas por el mismo vector de nodos y los mismos puntos de control) y se dibujan las dos curvas (en verde la polinomial y en azul la racional). Lo mismo para superficies. En el caso de las superficies, se dibujan las curvas del borde.

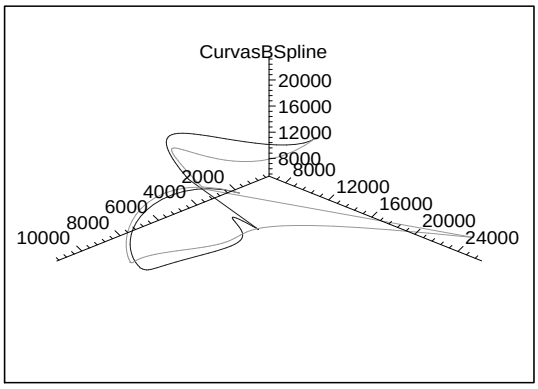
```
> read "simb.txt";
```

El orden de la curva es: 6

El vector de nodos de la curva: [0, 0, 0, 0, 0, 0, .2736069288,
.2108116634, .6488426571, .4016136423, 1., 1., 1., 1., 1., 1.]

Los puntos de control: [[2258.072848, 6915.017654, 10258.09613,
11288.06777, 12324.01982, 4780.115079, 10455.10969, 7644.316017,
10224.23333, 3574.225166], [7321.040000, 5146.080537, 8719.020747,
18066.01291, 11203.11796, 20686.11671, 18115.19923, 4390.282051,
15117.02733, 16814.31278], [5621.114094, 14674.08514, 10029.22167,
5363.058824, 11554.25094, 8844.187500, 16810.17783, 28899.24047,
11640.29302, 21899.20502]]

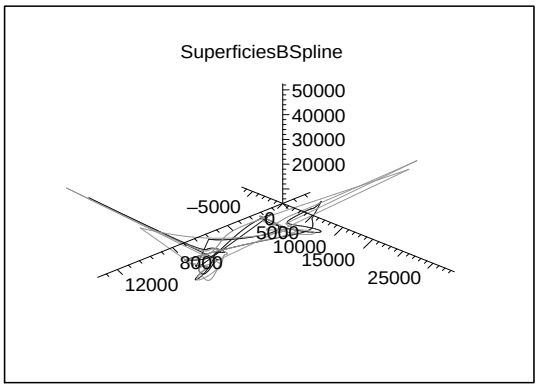
Los pesos: [.7451474921, .5353707958, .6137492977, .6851157637,
.2049666210, .4424820359, .9955525285, .8874027309, .3484910896,
.8492084257]



Los ordenes de la superficie: 6 5

Los vectores de nodos: [0, 0, 0, 0, 0, 0, .2736069288, .2108116634,
.6488426571, .4016136423, 1., 1., 1., 1., 1., 1.] [0, 0, 0, 0, 0,
.8306417752, .8324759978, .5217030071, .3990572283, .2997184002, 1.,
1., 1., 1., 1.]

Los puntos de control: [[[4202.137694, 7276.136439, 2851.564356,
3106.068966, 6068.030405, 2943.414966, 8027.056625, 3038.088235,
.....
.....
.7383023371, .6030972644], [.7511890176, .6714217746, .4584764261,
.9990596216, .8357042218, .8443048216, .7424006280, .9141641517,
.3912476101, .7474945914]]



5 Manipulación de curvas B-spline (polinomiales y racionales)

Se muestra en esta sección como Maple puede de forma muy eficiente realizar diversas operaciones con curvas B-spline bien polinomiales o racionales.

```
> read "manipBspline.txt";
```

Consideramos los siguientes parametros de una curva B-spline:

```
> ListaNodos:=[0.,0.,0.,0.,0.,0.25,0.5,0.75,1.,1.,1.,1.,1.];
> orden:=5; nrPuntos:=nops(ListaNodos)-orden;
> ListaPuntos:=[[0.,10.,10.,20.,30.,40.,25.,15.],[0.,20.,30.,35.,35.,25.
> ,5.,10.],[0.,5.,5.,10.,10.,20.,5.,15.]];
```

$$ListaNodos := [0, 0, 0, 0, 0, .25, .5, .75, 1., 1., 1., 1., 1.]$$

$$orden := 5$$

$$nrPuntos := 8$$

$$ListaPuntos := [[0, 10., 10., 20., 30., 40., 25., 15.], [0, 20., 30., 35., 35., 25., 5., 10.], [0, 5., 5., 10., 10., 20., 5., 15.]]$$

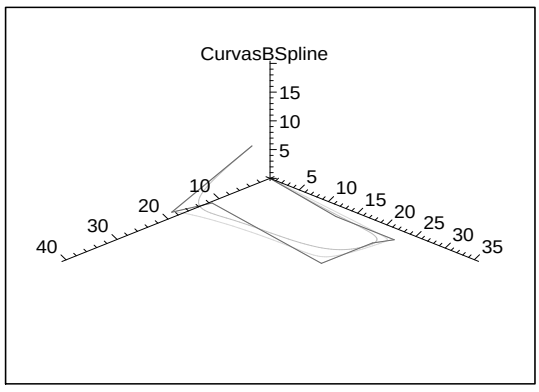
y el siguiente vector de pesos:

```
> ListaPesos:=[]:  for i from 1 to nrPuntos do
> ListaPesos:=[op(ListaPesos), evalf(rand()/10**12)] od:  ListaPesos;
```

$$[.6496557136, .06348724911, .6206061115, .1359878037, .7582417952, .2920246583, .8785076023, .0004649795500]$$

Se generan el poligono de control y las curvas, polinomial y racional, correspondientes a los datos anteriores:

```
> dibPol:=dibujoPol(orden,nrPuntos,ListaNodos,ListaPuntos,0.,1.,Col[nn(
> )],Col[nn()]) :
> dibRac:=dibujoRac(orden,nrPuntos,ListaNodos,ListaPesos,ListaPuntos,0.,
> 1.,Col[nn()],Col[nn()]):
> plots[display]([op(dibPol),op(dibRac)],
> axes=NORMAL,title='CurvasBSpline');
```



Se calculan las derivadas (primera y segunda) de las curvas, primero con el metodo de las funciones de base y luego con el metodo de los puntos de control en unos puntos generados aleatoriamente (despues de cada operacion aparece el tiempo correspondiente).

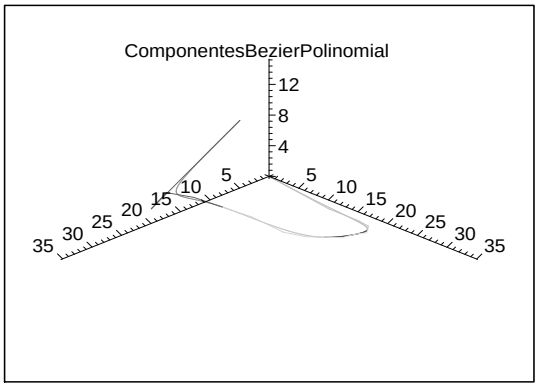
```
> for i from 1 to 5 do arg:=evalf(rand()/10**12); timp1:=time():
> derPolFB:=calcularDerivadasPolFB(arg,orden,nrPuntos,ListaNodos,ListaPu
> ntos); timp1:=time()-timp1; timp2:=time():
> derPolPC:=calcularDerivadasPolPC(arg,orden,nrPuntos,ListaNodos,ListaPu
> ntos,2); timp2:=time()-timp2; timp3:=time():
> derRacFB:=calcularDerivadasRacFB(arg,orden,nrPuntos,ListaNodos,ListaPe
> sos,ListaPuntos); timp3:=time()-timp3; timp4:=time():
> derRacPC:=calcularDerivadasRacPC(arg,orden,nrPuntos,ListaNodos,ListaPe
> sos,ListaPuntos,2); timp4:=time()-timp4; lprint(); lprint(`Argumento
> :`, arg); lprint(`Derivada polinomial FB: `,derPolFB); lprint(`Tiempo
> :`,timp1); lprint(`Derivada polinomial PC: `,derPolPC); lprint(`Tiempo
> :`,timp2); lprint(`Derivada racional FB: `,derRacFB); lprint(`Tiempo
> :`,timp3); lprint(`Derivada racional PC: `,derRacPC); lprint(`Tiempo
> :`,timp4);  od:


Argumento :   .8458933315

Derivada polinomial FB:    [[32.05687097, 17.50040607, 12.54996778],
[-54.43263906, -102.3150315, -37.93511739], [-749.8846345,
-41.0448997, -154.5126604]]

Tiempo :   .10e-1

Derivada polinomial PC:    [[32.05687097, 17.50040607, 12.54996778],
[-54.43263912, -102.3150317, -37.93511741], [-749.8846338,
-41.0448993, -154.5126601]]

Tiempo :   .20e-1

Derivada racional FB:    [[29.11414329, 13.78607835, 9.128659526],
[-37.30823064, -110.0511479, -37.68651421], [12.73743261, 732.2496156,
21.86782408]]

Tiempo :   .260

Derivada racional PC:    [[29.11414329, 13.78607835, 9.128659526],
[-37.30823073, -110.0511479, -37.68651425], [12.73743218, 732.2496156,
21.86782409]]

Tiempo :   .20e-1
```

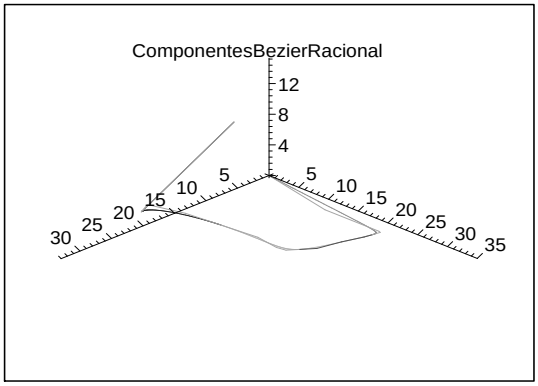
Se generan las componentes Bezier (polinomiales y racionales) de las curvas iniciales:

```
> BezierPol:=generarComponentesBezierPol(orden,nrPuntos,ListaNodos,List
> aPuntos):
> BezierRac:=generarComponentesBezierRac(orden,nrPuntos,ListaNodos,Lista
> Pesos,ListaPuntos):
```

Se dibujan todas las componentes Bezier de las curvas con los poligonos de control correspondiente:

```
> DibPol:=[]:  DibRac:=[]:  for i from 1 to nops(BezierPol[2]) do
> val1:=BezierPol[1][i]; val2:=BezierPol[1][i+1]; Nodos:=[]; for j from
> 1 to orden do Nodos:=[val1, op(Nodos), val2] od:
> PuntosPol:=BezierPol[2][i]; Pesos:=BezierRac[2][i][1];
> PuntosRac:=BezierRac[2][i][2]; nrP:=nops(Pesos);
> dibPol:=dibujoPol(orden,nrP,Nodos,PuntosPol,val1,val2,Col[nn()],
> Col[nn()]):
> dibRac:=dibujoRac(orden,nrP,Nodos,Pesos,PuntosRac,val1,val2,Col[nn()],
> Col[nn()]):  DibPol:=[op(DibPol), op(dibPol)]:  DibRac:=[op(DibRac),
> op(dibRac)]:  od:  plots[display]([op(DibPol)],axes=NORMAL,
> title='ComponentesBezierPolinomial');
> plots[display]([op(DibRac)],axes=NORMAL,
> title='ComponentesBezierRacional');
```

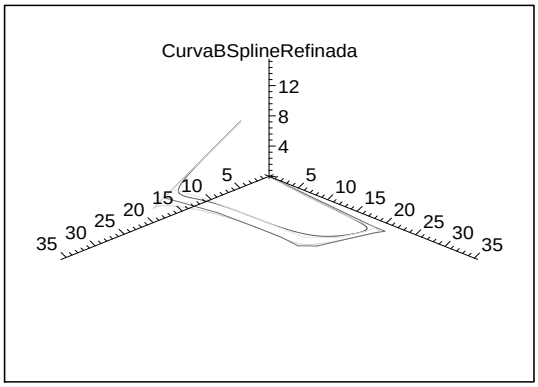




Se refina el vector de nodos utilizando los siguientes nodos y se dibujan de nuevo el poligono de control y las curvas:

```
> ListaNodosAnnadir:=[0.1,0.4,0.7,0.7,0.7];
> auxPol:=refinarListaNodosPol(orden,nrPuntos,ListaNodos,ListaPuntos,Lis
> taNodosAnnadir,3):
> auxRac:=refinarListaNodosRac(orden,nrPuntos,ListaNodos,ListaPesos,List
> aPuntos,ListaNodosAnnadir):
> Nodos:=auxPol[1]:  PuntosPol:=auxPol[2]:  nr:=nops(Nodos)-orden:
> dibPol:=dibujoPol(orden,nr,Nodos, PuntosPol,0.,1.,Col[nn()],Col[nn()])
> :
> PuntosRac:=auxRac[3]:  Pesos1:=auxRac[2]:
> dibRac:=dibujoRac(orden,nr,Nodos,Pesos1,PuntosRac,0.,1.,Col[nn()],Col[
> nn()]):
> plots[display]([op(dibPol),op(dibRac)],
> axes=NORMAL,title='CurvaBSplineRefinada');
```

$$ListaNodosAnnadir := [.1,\ .4,\ .7,\ .7,\ .7]$$



6 Manipulación de superficies B-spline (polinomiales y racionales)

Se muestra en esta sección como Maple puede de forma muy eficiente realizar diversas operaciones con superficies B-spline bien polinomiales o racionales.

```
> read "manipBsplineSup.txt";
```

Consideramos los siguientes parametros de una superficie B-spline polinomial:

```
> ListaNodosS:=[0.,0.,0.,0.,0.,0.,0.25,0.25,0.25,0.25,0.75,0.75,0.75,0.
> 75,1.,1.,1.,1.,1.,1.];
> ordenS:=6; nrPuntosS:=nops(ListaNodosS)-ordenS;
> ListaNodosT:=[0.,0.,0.,0.,0.,0.,0.3,0.3,0.3,0.3,0.6,0.6,0.6,0.6,1.,1.,
> 1.,1.,1.,1.];                                                ordenT:=6;
> nrPuntosT:=nops(ListaNodosT)-ordenT;
```

$$ListaNodosS := [0, 0, 0, 0, 0, 0, .25, .25, .25, .25, .75, .75, .75, .75, 1., 1., 1., 1., 1., 1.]$$

$$ordenS := 6$$

$$nrPuntosS := 14$$

$$ListaNodosT := [0, 0, 0, 0, 0, 0, .3, .3, .3, .3, .6, .6, .6, .6, 1., 1., 1., 1., 1., 1.]$$

$$ordenT := 6$$

$$nrPuntosT := 14$$

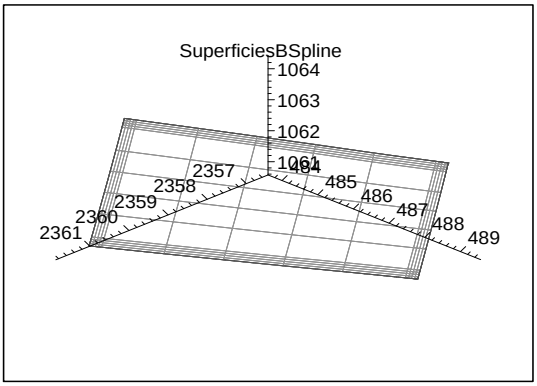
```
> fis:=fopen(`datos.txt`, READ):   ListaPuntosX:=[]:
> ListaPuntosY:=[]:  ListaPuntosZ:=[]:
> for i from 1 to nrPuntosT do
> ListaX:=[]; ListaY:=[]; ListaZ:=[];
> for j from 1 to nrPuntosS do
> ListaX:=[op(ListaX), fscanf(fis, %f)[1]];
> ListaY:=[op(ListaY), fscanf(fis, %f)[1]];
> ListaZ:=[op(ListaZ), fscanf(fis, %f)[1]];
> od:
> ListaPuntosX:=[op(ListaPuntosX), ListaX];
> ListaPuntosY:=[op(ListaPuntosY), ListaY];
> ListaPuntosZ:=[op(ListaPuntosZ), ListaZ];
> od:
> fclose(fis):  ListaPuntos:=[ListaPuntosX, ListaPuntosY,
> ListaPuntosZ]:
```

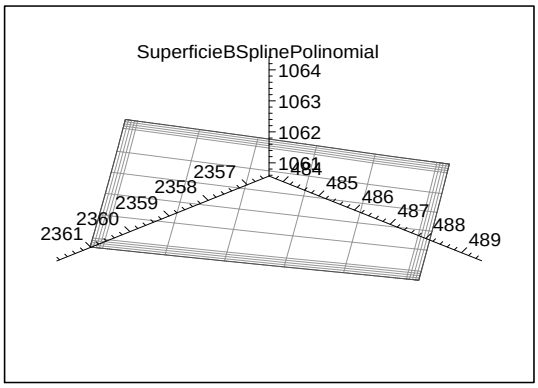
y la siquiente matriz de pesos:

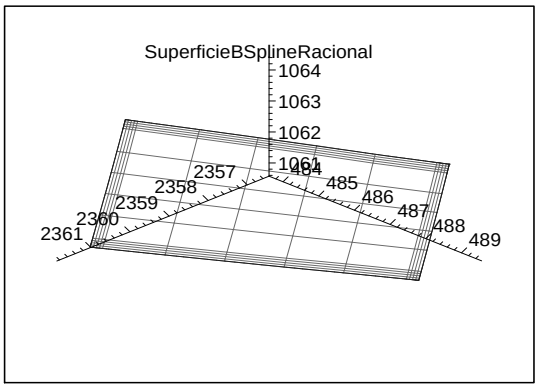
```
> ListaPesos:=[]:
> for i from 1 to nrPuntosT do Lista:=[]; for j from 1 to nrPuntosS do
> Lista:=[op(Lista), evalf(rand()/10**12)] od:
> ListaPesos:=[op(ListaPesos), Lista] od:  ListaPesos:
```

Generamos las dos superficies B-spline (polinomial y racional) y las dibujamos junto con su poligono de control.

```
> dibPol:=dibujoSupPol(ordenS,ordenT,nrPuntosS,nrPuntosT,ListaNodosS,Li
> staNodosT,ListaPuntos,0.,1.,0.,1.,'red','green'):
> dibRac:=dibujoSupRac(ordenS,ordenT,nrPuntosS,nrPuntosT,ListaNodosS,Lis
> taNodosT,ListaPesos,ListaPuntos,0.,1.,0.,1.,'blue','magenta'):
> plots[display]([op(dibPol), op(dibRac)],
> axes=NORMAL, title=`SuperficiesBSpline`);
> plots[display]([op(dibPol)], axes=NORMAL,
> title=`SuperficieBSplinePolinomial`); plots[display]([op(dibRac)],
> axes=NORMAL, title=`SuperficieBSplineRacional`);
```







Calculamos las derivadas (de orden total 2) de las superficies en 5 puntos generados aleatoriamente, utilizando los dos metodos: el metodo de las funciones de base y el metodo de los puntos de control.

En el caso del metodo de las funciones de base, los resultados aparecen en la pantalla en el siguiente orden: Der00,Der10,Der01,Der20,Der11,Der02.

En el caso del metodo de los puntos de control los resultados aparecen en el siguiente orden: Der00,Der01,Der02,Der10,Der11,Der20. Parte del output ha sido suprimido.

```
> Digits:=16:
> for i from 1 to 5 do argS:=evalf(rand()/10**12);
> argT:=evalf(rand()/10**12); timp3:=time();
> derPolFB:=calcularDerivadasPolSuperficieFB(argS,argT,ordenS,nrPuntosS,
> ordenT,nrPuntosT,ListaNodosS,ListaNodosT,ListaPuntos);
> timp3:=time()-timp3; timp4:=time();
> derPolPC:=calcularDerivadasPolSuperficiePC(argS,argT,ordenS,nrPuntosS,
> ListaNodosS,ordenT,nrPuntosT,ListaNodosT,ListaPuntos,2);
> timp4:=time()-timp4;  timp1:=time();
> derRacFB:=calcularDerivadasRacSuperficieFB(argS,argT,ordenS,nrPuntosS,
> ordenT,nrPuntosT,ListaNodosS,ListaNodosT,ListaPesos,ListaPuntos);
> timp1:=time()-timp1; timp2:=time();
> derRacPC:=calcularDerivadasRacSuperficiePC(argS,argT,ordenS,nrPuntosS,
> ListaNodosS,ordenT,nrPuntosT,ListaNodosT,ListaPesos,ListaPuntos,2);
> timp2:=time()-timp2; lprint(); lprint(`Argumetos:  `,argS,argT);
> lprint(`Derivadas Pol FB:`); for j from 1 to nops(derPolFB) do
> lprint(derPolFB[j]); od; lprint(`Tiempo`,timp3); lprint(`Derivadas Pol
> PC:`);  for j from 1 to nops(derPolPC) do lprint(derPolPC[j]); od;
> lprint(`Tiempo:  `,timp4); lprint(`Derivadas Rac FB: `);
> for j from 1 to nops(derRacFB) do lprint(derRacFB[j]); od:
> lprint(`Tiempo:  `,timp1); lprint(`Derivadas Rac PC: `);
> for j from 1 to nops(derRacPC) do lprint(derRacPC[j]); od:
> lprint(`Tiempo:  `,timp2); od:
> Digits:=10:


Argumetos:    .2343384383590000   .4638661136260000

Derivadas Pol FB:

[2359.518940493465, 487.0311374040819, 1061.277007256592]

[-1.6944287778418, -.87185075402776, 2.7732192030585]

[-10.7077915869695, 14.11671900866031, -1.9524893749358]

[-18.621023679972, -9.5753667293827, 30.463574828936]

[.601405722411, .201018274149, -.807455668135]

[6.569414433638, -9.286303630796, 1.756269732548]

Tiempo   .110

Derivadas Pol PC:

[[2359.518940493467, 487.0311374040819, 1061.277007256593],
[-10.70779158696980, 14.11671900866196, -1.952489374935699],
[6.569414433575617, -9.286303630814925, 1.756269732523466]]

[[-1.694428777840756, -.8718507540276519, 2.773219203059553],
[.6014057224017089, .2010182741482566, -.8074556681440215]]
```

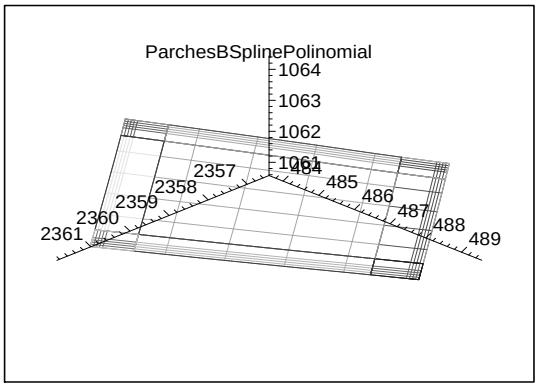
```
[[-18.62102368007849, -9.575366729395748, 30.46357482892405]]

Tiempo:    .520
```

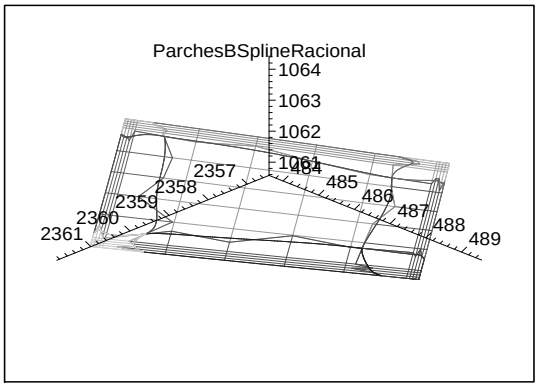
Descomponemos las superficies B-spline en sus componentes (segmentos) Bezier:

```
> BezierPol:=generarParchesBezierPol(ordenS,ordenT,nrPuntosS,nrPuntosT,
> ListaNodosS,ListaNodosT,ListaPuntos):
> BezierRac:=generarParchesBezierRac(ordenS,ordenT,nrPuntosS,nrPuntosT,L
> istaNodosS,ListaNodosT,ListaPesos,ListaPuntos):
```

y los dibujamos junto con los poligonos de control correspondientes:

```
> nr1:=nops(BezierPol[1])-1:  nr2:=nops(BezierPol[2])-1:
> NodosA:=BezierPol[1]:  NodosB:=BezierPol[2]:
> DibPol:=[]:  DibRac:=[]:  DibPol1:=[]:  DibRac1:=[]:
> for i from 1 to nr2 do NodosT:=[]:  for j from 1 to ordenT do
> NodosT:=[NodosB[i],op(NodosT),NodosB[i+1]] od:
> for k from 1 to nr1 do NodosS:=[]:  for j from 1 to ordenS do
> NodosS:=[NodosA[k],op(NodosS),NodosA[k+1]] od:
> PuntosPol:=BezierPol[3][(i-1)*nr1+k];dibPol:=dibujoSupPol(ordenS,orden
> T,ordenS,ordenT,NodosS,NodosT,PuntosPol,NodosA[k],NodosA[k+1],NodosB[i
> ],NodosB[i+1],Col[nn()],Col[nn()]):  DibPol:=[op(DibPol), op(dibPol)]:
> DibPol1:=[op(DibPol1), dibPol]:    val:=BezierRac[3][(i-1)*nr1+k];
> PuntosRac:=[op(1..3,val)];Pesos:=val[4];dibRac:=dibujoSupRac(ordenS,or
> denT,ordenS,ordenT,NodosS,NodosT,Pesos,PuntosRac,NodosA[k],NodosA[k+1]
> ,NodosB[i],NodosB[i+1],Col[nn()],Col[nn()]):  DibRac:=[op(DibRac),
> op(dibRac)]:  DibRac1:=[op(DibRac1), dibRac] ; od:  od:
> plots[display]([op(DibPol)], axes=NORMAL,
> title=`ParchesBSplinePolinomial`);
> plots[display]([op(DibRac)], axes=NORMAL,
> title=`ParchesBSplineRacional`); for i from 1 to nr1*nr2 do
> plots[display]([op(DibPol1[i]),op(DibRac1[i])], axes=NORMAL,
> title=`ParcheBSpline`)  od;
```





Vamos a refinar los vectores de nodos, annadiendo los siguientes vectores:

```
> NodosAnnadirS:=[0.1,0.5,0.9]; NodosAnnadirT:=[0.2,0.5,0.8];
```

$$NodosAnnadirS := [.1,\ .5,\ .9]$$

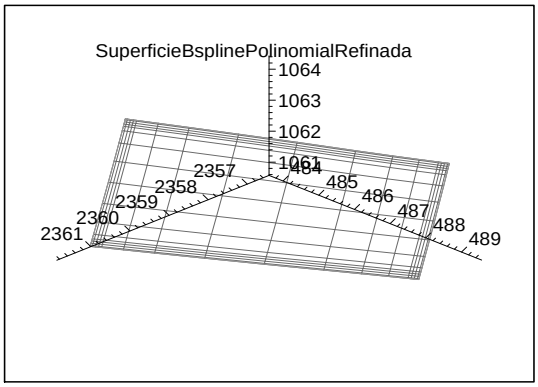
$$NodosAnnadirT := [.2,\ .5,\ .8]$$

Insertamos los nodos primero en la dirteccion 'S' y luego en la direccion 'T'

```
> aux1:=refinarNodosSuperfPol("S",ordenS,ordenT,nrPuntosS,nrPuntosT,Lis
> taNodosS,ListaNodosT,ListaPuntos,NodosAnnadirS,3):  NodosS:=aux1[1]:
> NodosT:=aux1[2]:  nrS:=nops(NodosS)-ordenS: nrT:=nops(NodosT)-ordenT:
> aux2:=refinarNodosSuperfPol("T",ordenS,ordenT,nrS,nrT,NodosS,NodosT,au
> x1[3],NodosAnnadirT,3):  NodosS:=aux2[1]:  NodosT:=aux2[2]:
> PuntosPol:=aux2[3]:  nrS:=nops(NodosS)-ordenS:
> nrT:=nops(NodosT)-ordenT:
```

y dibujamos la 'nueva' superficie:

```
> dibPol:=dibujoSupPol(ordenS,ordenT,nrS,nrT,NodosS,NodosT,PuntosPol,0.
> ,1.,0.,1.,Col[nn()],Col[nn()]):    plots[display]([op(dibPol)],
> axes=NORMAL, title=`SuperficieBsplinePolinomialRefinada`);
```



Hacemos lo mismo para la superficie B-spline racional:

```
> aux1:=refinarNodosSuperfRac("S",ordenS,ordenT,nrPuntosS,nrPuntosT,Lis
> taNodosS,ListaNodosT,ListaPesos,ListaPuntos,NodosAnnadirS):
> NodosS:=aux1[1]:  NodosT:=aux1[2]:  nrS:=nops(NodosS)-ordenS:
> nrT:=nops(NodosT)-ordenT:
> aux2:=refinarNodosSuperfRac("T",ordenS,ordenT,nrS,nrT,NodosS,NodosT,au
> x1[3],aux1[4],NodosAnnadirT):
> NodosS:=aux2[1]:  NodosT:=aux2[2]:  Pesos:=aux2[3]:  PuntosRac:=aux2[4]:
> nrS:=nops(NodosS)-ordenS: nrT:=nops(NodosT)-ordenT:
> dibRac1:=dibujoSupRac(ordenS,ordenT,nrS,nrT,NodosS,NodosT,Pesos,Puntos
> Rac,0.,1.,0.,1.,Col[nn()],Col[nn()]):
> plots[display]([op(dibRac1)], axes=NORMAL,
> title=`SuperficieBSplineRacionalRefinada`);
```