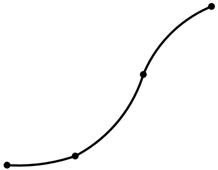
CM2202: Scientific Computing and Multimedia Applications
Lab Class Week 9

School of Computer Science & Informatics

Fitting and Interpolation Using Parametric Polynomials



- Choose a value of t which corresponds to each given point, thus determining the order in which points occur on the curve.
- Chosen values of t and corresponding values of x and y substituted at each point, give a set of linear simultaneous equations to solve for parameters, a_i, b_i, c_i *etc.*
- If the order of the curve (highest power of t) is one less than the number of points (3 for quadratic, 4 for cubic *etc.* then the simultaneous equations can be solved.

The above procedure (interpolation through points) is called **Lagrangian Interpolation**. Lagrangian interpolation demo code

CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

Lagrangian Interpolation

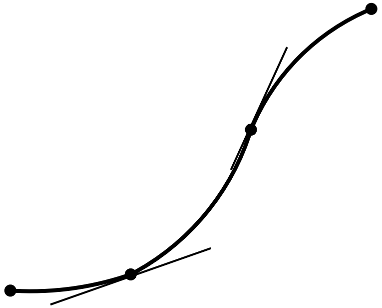
lagrangian_demo.m

```
%%%% Demo to illustrate Lagrangian Interpolation Code
close all;
clear all;

% Define Lagrangian Polynomial Values

x = [1 3 5 7];  % Polynomial Values at x = 1, 3, 5, 7

y = [2 1 8 4];  % y values for x = 1, 3, 5, 7

% Compute a Cubic Lagrangian Polynomial
[a b c d] = lagrangian_cubic_interpolate(x,y)

% Now PLOT THE POLYNOMIAL
x = 1:0.05:7; % Step through the clamped x values at some step

% Compute y Values for given cubic from a,b, c and d
[m n] = size(x)
A = [x.*x.*x; x.*x; x; ones(1,n)]';
y = A*[a b c d]';

% Plot the cubic
plot(x,y);
shg; % Show the current graphic
```

Use `grid on` to read the positions more easily.

Hermite Interpolation

Here we need to introduce and fulfil some slope constraints on the parametric polynomial.



- Slope means gradient or tangent at a point here.

Hermite Interpolation

- We need to compute the **partial derivatives** of the parametric polynomial. To this we differentiate each equation in x and y with respect to t
  For example for a cubic:

$$
\begin{aligned}
x &= a_1 + b_1 t + c_1 t^2 + d_1 t^3 \\
y &= a_2 + b_2 t + c_2 t^2 + d_2 t^3
\end{aligned}
$$

  We get the derivatives:

$$
\begin{aligned}
\frac{\partial x}{\partial t} &= b_1 + 2c_1 t + 3d_1 t^2 \\
\frac{\partial y}{\partial t} &= b_2 + 2c_2 t + 3d_2 t^2
\end{aligned}
$$

Hermite Interpolation

Some points to note:

- Gradients at each point need to estimated and then they can be substituted into the above equations and solved **together** with the original (Lagrangian) point
- It is not necessary to have slope constraints at every point — position and slope constraints can be mixed as required (so long as we have enough to satisfy the simultaneous
- If the points are spread evenly then the point can be parameterised at equal intervals of t.
- Setting start $t = 0$ and end $t = 1$ and having proportional values of t for unequal steps of t is a common approach.
- In Hermite interpolation there are no unique values for $\frac{\partial x}{\partial t}$ and $\frac{\partial y}{\partial t}$ for a required $\frac{dx}{dy}$, only the ratio $\frac{\partial x}{\partial t} / \frac{\partial y}{\partial t}$ must correspond. This can introduce some unwanted results.
- As the order of the curves becomes higher, undesired oscillations, **waviness**, tends to occur. Higher than order 5 or 6 is not common.
- There are more elaborate parametric curve representation — Bézier curves, Spline curves.

MATLAB Hermite spline interpolation example, hermite interpolation demo code

Hermite Interpolation (Explicit)

Explicit cubic polynomial: hermite_demo.m (use `grid on` to show the grid).

```
%%%% Demo to illustrate Hermite Interpolation Code
close all;
clear all;

% Define Hermite Polynomial Values
x = [1 3];  % Polynomial Values at x = 1 and 3
dx = [1 3]; % Derivative Values at x = 1 and 3

y = [2 1];  % y values for x = 1 and 3
dy = [1 2] % Derivative values for dx = 1 and 3

% Compute a CUbic Hermite Polynomial
[a b c d] = hermite_cubic_interpolate(x,y,dx,dy);

% Now PLOT THE POLYNOMIAL
x = 1:0.05:3 % Step through the clamped x values at some step

% Compute y Values for given cubic from a,b, c and d
[m n] = size(x)
A = [x.*x.*x; x.*x; x; ones(1,n)]';
y = A*[a b c d]';

% Plot the cubic
plot(x,y);
shg; % Show the current graphic
```

Hermite Interpolation (Parametric)

Parametric cubic polynomial: hermite_parametric_demo.m (use `grid on` to show the grid).

```
%%%% Demo to illustrate Hermite Interpolation Code
close all;
clear all;

% Define Hermite Polynomial Values
tx = [1 2];
x = [1 3];  % Polynomial Values at t = 1 and 2
ty = [1 2];
y = [2 1];  % y values for t = 1 and 2
tdy = [1 2]; % Derivative Values at t = 1 and 2
dydx = [1 2]; % Derivative values for dx = 1 and 3
dydxratio = 1;

% Compute a Cubic Hermite Polynomial
[a1,b1,c1,d1,a2,b2,c2,d2] = hermite_parametric_cubic_interpolate(tx,x,ty,y,tdy,dydx,dydxratio)

% Now PLOT THE POLYNOMIAL
t = 1:0.025:2; % Step through the clamped x values at some step
% Compute y Values for given cubic from a,b, c and d
[m n] = size(t);
A = [t.*t.*t; t.*t; t; ones(1,n)]';
x = A*[a1 b1 c1 d1]';
y = A*[a2 b2 c2 d2]';
% Plot the cubic
plot(x,y);
shg; % Show the current graphic
```

Plot 3D lines in MATLAB

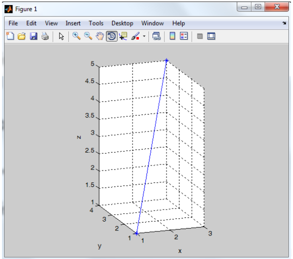
To plot a line segment with end points (x_1, y_1, z_1) and (x_2, y_2, z_2), you can use `plot3([x1 x2], [y1 y2], [z1 z2]);` (similar to `plot` in 2D – see `help plot3`).

Example: To plot a line segment from $(1, 1, 1)$ to $(3, 4, 5)$:

```
>> plot3([1 3], [1 4], [1 5], '*-');
```

To make the 3D line more clearly visible, you may enable the grid and add labels to x-/y-/z-axes.

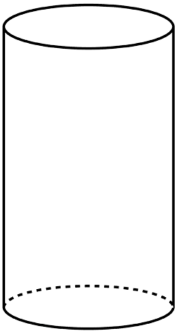
```
>> grid on; axis equal; xlabel('x'); ylabel('y'); zlabel('z');
```

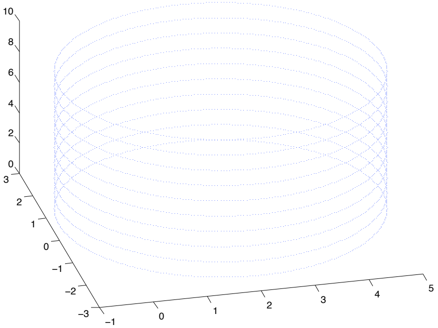


Parametric Surface: Cylinder

For example, a cylindrical may be represented in parametric form as

$$x = x_0 + r\cos u \qquad y = y_0 + r\sin u \qquad z = z_0 + v.$$





Parametric Surface: Cylinder (MATLAB Code)

The MATLAB code to plot the cylinder figure is cyl_plot.m

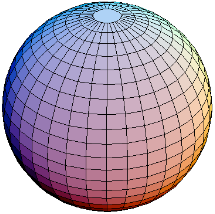
```
p0 = [2,0,0] % x_0, y_0, z_0
r = 3; %radius

n = 360;

hold on;
for v = 1:10
for u = 1:360
theta = ( 2.0 * pi * ( u - 1 ) ) / n;
x = p0(1) + r * cos(theta);
y = p0(2) + r * sin(theta);
z = p0(3) + v;

plot3(x,y,z);
end
end
```

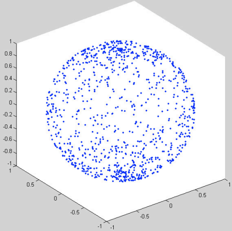
CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

Parametric Surface: Sphere

A sphere is represented in parametric form as

$$x = x_c + r\sin(u)\sin(v) \qquad y = y_c + r\cos(u)\sin(v) \qquad z = z_c + r\cos(v)$$





MATLAB code to produce a parametric sphere is at HyperSphere.m (see `help HyperSphere` for examples).