

In this project you will construct an adaptive cubic spline function to approximate a given function to a specified accuracy. You will also learn how to plot planar curves. In addition you will enter an exciting competition, whose winner will receive a prize of great value.

Adaptive cubic splines

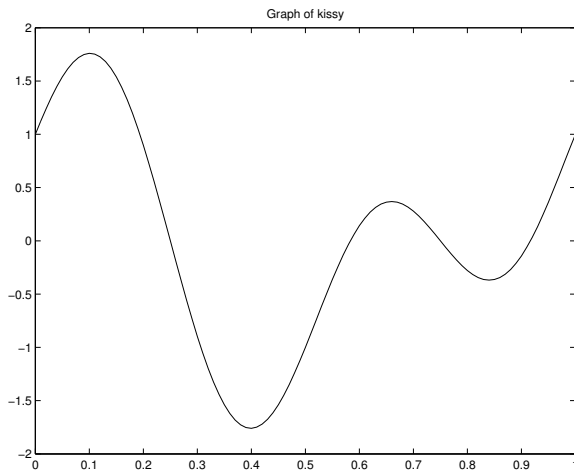
We have seen how to use the Matlab function `spline` to interpolate a set of data at a grid of knots. We have also seen that, under certain regularity conditions, as the maximum distance between consecutive knots goes to zero, the approximation converges to the underlying function. This statement is of course true, but it is a little misleading.

Consider the function

```
function y = kissy(t)
    y = sin(4*pi*t) + cos(2*pi*t);
return
```

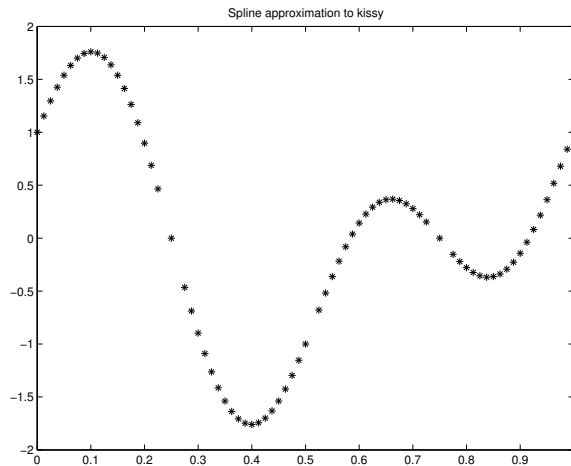
 (1)

On the interval $[0, 1]$ it has the graph



If we wish to approximate by a cubic spline, the convergence theory suggests we take a mesh of equally spaced knots in $[0, 1]$, compute the spline and evaluate it on a finer mesh. If the approximation is not satisfactory, compute a new equally spaced mesh with more points and try again until the desired accuracy is achieved. While this procedure will work, it may produce a finer mesh than is really necessary.

Consider the following graph of a spline approximation to `kissy`.



(2)

which is accurate to 10^{-5} . The asterisks indicate the placement of the knots. What we see here is that the knots are tightly spaced where the graph of `kissy` is highly curved and they are loosely spaced where the graph is less curved. If we used an equally spaced mesh, we would be interpolating at points that are not needed.

Your assignment is to write a function

```
function [xx, pp] = acs(x, func, tol)
```

(`acs` stands for `adaptive cubic spline`). This function starts with a course mesh `x` and attempts to refine it so that

```
abs(feval(func, t) - ppval(pp,t)) <= tol, xx(1) <= t <= xx(n),
```

where `n` is the number of points in the refined grid `pp`. For example, the graph (2) was generated by the matlab statements

```
tol = 1e-5;
[ty, py] = acs(0:.1:1, 'kissy', tol);
plot(ty, ppval(py,ty), '*');
```

(3)

The basic algorithm goes as follows.

1. Compute the spline `p` for `(x, feval(func, x))`.
2. Compute the array of midpoints between successive knots and the the values of `p` at the midpoints.
3. Construct another mesh `xt` by adding all midpoints `m` to mesh `x` for which

```
abs(feval(func, m) - ppval(p,m)) > tol
```

4. If \mathbf{x} at \mathbf{xt} are different, set $\mathbf{x} = \mathbf{xt}$ and repeat the above. Otherwise return \mathbf{x} and \mathbf{p}

To test your function, run the script (3) for

```
tol = [1e-4, 1e-5, 1e-6]
```

Print out the three plot and submit them with a listing of your function `acs`. [Hint: You can print out a plot from a script using the `matlab` print command. for example,

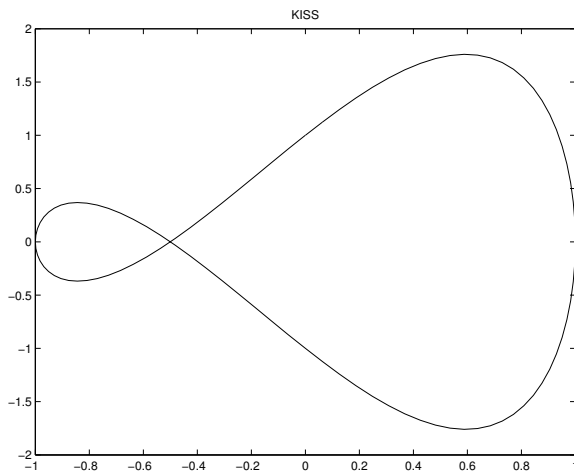
```
print(gcf, '-deps', 'kissyplot')
```

will print the current figure as an eps (encapsulated postscript) file, which you may then send to any postscript printer.]

The function `acs` is not foolproof. How would you construct a function on which it will not work? Note, you don't need to give formulas. A hand drawn picture will do. Submit your written answer with the other material. [Hint: Consider the union of the original mesh \mathbf{x} and **all** of the midpoints you constructed in the course of running `acs`, whether or not the midpoints were incorporated into the final mesh.]

Planar curves

Plotting a variable y and a function of another variable x , can never produce a curve like a complete circle or a figure eight. For example, here is a curve called `kiss`.



(It is called `kiss` because if you rotate it ninety degrees clockwise it looks a little like a Hersey kiss.) How to you plot such things. The trick is to represent the coordinates of `kiss` by two functions $x(t)$ and $y(t)$. We have already defined $y(t)$ by the function `kissy` in (1). The corresponding function $x(t)$ is given by

```
function x = kissx(t)
x = sin(2*pi*t)
```

The plot was then generated by

```
t = 0:.01:1;
plot(kissx(t),kissy(t));
```

For this part of your assignment, write a script to run `acs` on `kissx` and `kissy` to get two spline approximations. Take `tol=1e-5`. Plot the values for the each spline as in (2). Also plot the spline approximation of `kiss` on a mesh of `0:0.01:1`. Hand in your script and the three plots.

A contest

There are many planar curves like `kiss`, some of them strikingly beautiful. Try your hand at producing a new esthetically pleasing curve. Our TA will judge which is the best. The winner gets a free double-scoop ice cream cone at the University Visitor's Center. You even get to choose the flavor.