

MATH 128A, SUMMER 2009: PROGRAMMING ASSIGNMENT 1

In this assignment, we will use Newton's divided difference algorithm to approximate the function $\cos(x)$ by a Lagrange polynomial in divided-difference form.

There are three steps involved. The first step is to write a general function that computes the divided-difference coefficients $a_k = f[x_0, \dots, x_k]$ of the Lagrange interpolating polynomial, given as input a vector containing x_0, \dots, x_n . The second step will be to evaluate the Lagrange interpolating polynomial (in its divided-difference form $P_n(x) = \sum_{k=0}^n f[x_0, \dots, x_k] \prod_{j=0}^k (x - x_j)$) at a few sample points. The third step will be to use the MATLAB plot function to graph the interpolating polynomial for the Cosine function, together with the Cosine function on the same set of axes over an appropriately-chosen interval.

We begin by giving some example code. The following recursive MATLAB function will evaluate $a_0 + a_1(x - x_0) + \dots + a_n(x - x_0) \dots (x - x_{n-1})$, given as input vectors containing a_0, \dots, a_n and x_0, \dots, x_n .

When $n > 0$, it works by evaluating

$$a_0 + (x - x_0)[a_1 + a_2(x - x_1) + \dots + a_n(x - x_1) \dots (x - x_{n-1})].$$

```
function y=newton_eval(x,ai,xi)
if length(xi)==1
    y = ai(1);
else
    y = ai(1) + (x-xi(1)) .* newton_eval(x,ai(2:end),xi(2:end));
end
```

1. Write a recursive function `y=divided_difference(f, xi)`, which computes $f[xi(1), \dots, xi(end)]$ directly from its definition:

$$f[x_0] = f(x_0)$$

$$f[x_0, \dots, x_n] = \frac{f[x_1, \dots, x_n] - f[x_0, \dots, x_{n-1}]}{x_n - x_0}, \text{ if } n > 0$$

Note: The algorithm given in section 3.2 is more efficient.

2. Create a set of 6 equally-spaced nodes in the interval $[0, 10]$:

```
xi = linspace(0, 10, 6);
```

Calculate a vector `ai`, where `ai(k) = f[xi(1), ..., xi(k)]`.

Use `newton_eval` to evaluate the interpolating polynomial at $x = \pi/3$, $x = \pi/2$, and $x = \pi$. Give these estimates, and the absolute error, accurate to 6 significant digits.

3. Plot $y = \cos(x)$ and the interpolating polynomial $y = P(x)$ on the same set of axes:

```
xdata = linspace(0, 10);
ydata1 = newton_eval(xdata, ai, xi);
ydata2 = cos(xdata);
plot(xdata, ydata1, xdata, ydata2)
```

Your report should contain your code for `divided_difference`, the MATLAB commands you used, the numerical results from problem 2, and the graph from problem 3.