Babbage

A brief explanation of the method of differences, used in Charles Babbage's Difference Engine #2

Original Mathematica code by Ed Theren, rewritten by Scott Centoni.

The Difference Engines were designs for large machines to automatically print out tables of mathematical functions by extrapolation from a small number of points, using the method of differences.

---

initialization

```
CleanSlate[]
```

■ imports

```
Needs["Graphics`Graphics`"]
```

■ prologs

Automatically display matrices in MatrixForm.

```
$PrePrint = If[MatrixQ[#], MatrixForm[#], #] &;
```

Improve the appearance of text in graphs.

```
$TextStyle = {FontFamily → "Palatino", FontSize → 12} ; $FormatType = TraditionalForm ;
```

■ misc

Print non-matrix 2D tables in a format that keeps the items centered on each line.

```
CheckerInterleave[x_] := Most@Flatten[{#, ""} & /@ x, 1]

CheckerPad[n_][x_] := Module[{
   padding = Table["", {k, n - Length[x]}]
  },
  Join[padding, CheckerInterleave[x], padding ]
 ]
```

```
CheckerForm[x_] := Module[{
   width = Max[Length /@ x]
   (*,strlen=Max@Map[Length@ToString,x,{2}]*)
  },
  TableForm[
   CheckerPad[width] /@ x,
   TableSpacing → {4, 1}
  ]
 ]
```

Pascal's triangle

```
Table[Binomial[n, k], {n, 0, 8}, {k, 0, n}]
```

```
{{1}, {1, 1}, {1, 2, 1}, {1, 3, 3, 1}, {1, 4, 6, 4, 1}, {1, 5, 10, 10, 5, 1},
 {1, 6, 15, 20, 15, 6, 1}, {1, 7, 21, 35, 35, 21, 7, 1}, {1, 8, 28, 56, 70, 56, 28, 8, 1}}
```

```
CheckerForm[Table[Binomial[n, k], {n, 0, 8}, {k, 0, n}]]
```

```
                          1

                      1       1

                  1       2       1

              1       3       3       1

          1       4       6       4       1

      1       5       10      10      5       1

  1       6       15      20      15      6       1

1   7       21      35      35      21      7   1

1   8   28      56      70      56      28  8   1
```

method of differences

Perhaps the best way of thinking of the Difference Engine is as a machine for extrapolating tables of numbers. For instance, if we have a sequence of numbers 0,1,8,27,64, we recognize this as a list of the first few cubes. But if we did not make use of this information, how could we predict the next number? One way to approach this is to look at the successive differences from one item to the next:

```
Differences[x_] := Rest[x] - Most[x]
```

```
seq = {0, 1, 8, 27, 64, 125, 216};
```

```
Differences@%
```

```
{1, 7, 19, 37, 61, 91}
```

Hmm, not quite obvious what the pattern is, so let's repeat the process

```
Differences@%
```

```
{6, 12, 18, 24, 30}
```

Aha...well, we can predict the next item in this sequence, but let's do it again anyway

```
Differences@%
```

```
{6, 6, 6, 6}
```

```
Differences@%
```

```
{0, 0, 0}
```

We can see the pattern more clearly by showing all the differences together.

```
dat = NestList[Differences, seq, 5]
```

```
{{0, 1, 8, 27, 64, 125, 216}, {1, 7, 19, 37, 61, 91},
 {6, 12, 18, 24, 30}, {6, 6, 6, 6}, {0, 0, 0}, {0, 0}}
```

```
CheckerForm[dat]
```

```
0     1     8       27      64      125      216

   1     7      19      37      61       91

      6     12      18      24      30

         6      6       6       6

            0       0       0

                0       0
```

Of course, all the higher-order differences are exactly zero, so that taking the first item in each list of differences forms a new sequence, 0, 1, 6, 6, 0, ...

Now, with a bit of thought, it should be evident that this process can be reversed: if we have the sequence of coefficients, we can generate the sequence of values. That is precisely what the Difference Engine does: One inputs the difference coefficients, and cranks the machine to repeatedly add them up to obtain the values. Matrix multiplication is a useful way to express this process (though it was not developed until after the Difference Engine was designed).

```
crm[x_] := crm[x] = Table[If[0 ≤ j - i < 2, 1, 0], {i, x}, {j, x}]
```

```
cri[x_] := cri[x] = Table[If[i ≤ j, (-1)^(i+j), 0], {i, x}, {j, x}]
```

This is the matrix representing cranking through a Difference Engine with 6 wheels:

```
crm[6]
```

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

This is the matrix that represents cranking that machine backward:

```
cri[6]
```

$$\begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

If we do first one, then the other, then we get the identity matrix

```
crm[6].cri[6]
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
Crank[x_] := crm[Length[x]].x

ReverseCrank[x_] := cri[Length[x]].x
```

This basically cranks the machine once:

```
Crank[{0, 1, 6, 6}]
```

```
{1, 7, 12, 6}
```

That is, 0+1=1, 1+6=7, 6+6=12.

Let's crank the machine again:

```
Crank@%
```

```
{8, 19, 18, 6}
```

So 1+7=8, 7+12=19, 12+6=18.

Again:

```
Crank@%
```

```
{27, 37, 24, 6}
```

Or, better yet, several steps shown in sequence:

```
NestList[Crank, {0, 1, 6, 6, 0}, 7]
```

$$\begin{pmatrix} 0 & 1 & 6 & 6 & 0 \\ 1 & 7 & 12 & 6 & 0 \\ 8 & 19 & 18 & 6 & 0 \\ 27 & 37 & 24 & 6 & 0 \\ 64 & 61 & 30 & 6 & 0 \\ 125 & 91 & 36 & 6 & 0 \\ 216 & 127 & 42 & 6 & 0 \\ 343 & 169 & 48 & 6 & 0 \end{pmatrix}$$

And we see that the first column has reproduced our table of function values, plus one more--and we could keep on cranking the machine as long as we wanted. In most cases, we're only concerned with the values taken on sequentially by the register shown in the first column.

```
CrankTimes[regs_, n_] := First /@ NestList[Crank, regs, n]
```

```
CrankTimes[{0, 1, 6, 6, 0}, 20]
```

```
{0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000,
 1331, 1728, 2197, 2744, 3375, 4096, 4913, 5832, 6859, 8000}
```

```
CrankThrough[regs_] := CrankTimes[regs, Length[regs] - 1]
```

```
CrankThrough[{0, 1, 6, 6, 0}]
```

```
{0, 1, 8, 27, 64}
```

using the machine to crank out its own coefficients

There is a trick to use the machine to generate its own coefficients from a table of regularly-spaced values. If you reverse the sign of every other item in the sequence before and after cranking through the list, it just churns out the coefficients.

```
Alternate[x_] := x (-1)^Range[0,Length[x]-1]
```

```
DeCrankThrough[regs_] := Alternate@CrankThrough@Alternate@regs
```

```
DeCrankThrough[{0, 1, 8, 27, 64}]
```

```
{0, 1, 6, 6, 0}
```

If we input n numbers as function values, we can output n numbers as coefficients without making any assumptions about higher-order terms. If we output more than n numbers, we assume that the higher-order differences are all zero.

Let's look at what happens in the general case:

```
Array[a, 5]
```

```
{a[1], a[2], a[3], a[4], a[5]}
```

```
DeCrankThrough[Array[a, 5]] // TableForm
```

```
a[1]
-a[1] + a[2]
a[1] - 2 a[2] + a[3]
-a[1] + 3 a[2] - 3 a[3] + a[4]
a[1] - 4 a[2] + 6 a[3] - 4 a[4] + a[5]
```

This is very similar to the definition of a derivative, but note that the derivatives of different orders are being approximated at different points. We do recover the original points by cranking through the machine in the normal way:

```
CrankThrough@%
```

```
{a[1], a[2], a[3], a[4], a[5]}
```

scaling the step size

The technique of using the Difference Engine to generate its own coefficients from a sequence of equally-spaced values is of limited practical benefit. It's more useful if we can adjust the coefficients afterward to produce a more finely-spaced sequence.

To begin with, we consider the coefficients that correspond to the simple integer powers, 1, x, x^2, x^3, ...

```
polycoeff[0] = {1};
```

```
polycoeff[n_] := DeCrankThrough[Table[k^n, {k, 0, n}]]
```

```
CheckerForm[Table[polycoeff[k], {k, 0, 8}]]
```

```
                                    1
                                0       1
                            0       1       2
                        0       1       6       6
                    0       1       14      36      24
                0       1       30      150     240     120
            0       1       62      540     1560    1800    720
        0       1       126     1806    8400    16800   15120   504
    0       1       254     5796    40824   126000  191520  141120
```

We'll try the cube function again, but this time sample it with a step of two instead of one.

```
func = x^3;

Table[func, {x, 0, 14}]

{0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, 1728, 2197, 2744}

dat = Table[func, {x, 0, 14, 2}]

{0, 8, 64, 216, 512, 1000, 1728, 2744}

coe = DeCrankThrough[dat]

{0, 8, 48, 48, 0, 0, 0, 0}
```

We see that instead of the coefficient list {0,1,6,6,0,...} we got before, we get larger values. But we can't just multiply the coefficients by a constant to change the step size.

It turns out that we can multiply the coefficient list by a matrix that will scale the step. To scale a 7th-order coefficient list by one half, we multiply it on the left by

$$\text{halve} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{1}{8} & \frac{1}{16} & -\frac{5}{128} & \frac{7}{256} & -\frac{21}{1024} & \frac{33}{2048} \\ 0 & 0 & \frac{1}{4} & -\frac{1}{8} & \frac{5}{64} & -\frac{7}{128} & \frac{21}{512} & -\frac{33}{1024} \\ 0 & 0 & 0 & \frac{1}{8} & -\frac{3}{32} & \frac{9}{128} & -\frac{7}{128} & \frac{45}{1024} \\ 0 & 0 & 0 & 0 & \frac{1}{16} & -\frac{1}{16} & \frac{7}{128} & -\frac{3}{64} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{32} & -\frac{5}{128} & \frac{5}{128} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{64} & -\frac{3}{128} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{128} \end{pmatrix};$$

```
halve.coe

{0, 1, 6, 6, 0, 0, 0, 0}
```

This recovers the coefficient list we saw before, and the same technique works with any function.

```
CrankThrough[coe]

{0, 8, 64, 216, 512, 1000, 1728, 2744}

CrankThrough[halve.coe]

{0, 1, 8, 27, 64, 125, 216, 343}
```

We can repeat this to get a step size of one half:

```
halve.halve.coe
```

$$\left\{0, \frac{1}{8}, \frac{3}{4}, \frac{3}{4}, 0, 0, 0, 0\right\}$$

```
CrankThrough[halve.halve.coe]
```

$$\left\{0, \frac{1}{8}, 1, \frac{27}{8}, 8, \frac{125}{8}, 27, \frac{343}{8}\right\}$$

```
N@%
```

```
{0., 0.125, 1., 3.375, 8., 15.625, 27., 42.875}
```

The general scaling matrix (of the appropriate order) is a function of the scaling factor r.

```
CoeffMat[n_] := Transpose@Table[PadRight[polycoeff[k], n + 1], {k, 0, n}]

scalmat[r_, n_] := CoeffMat[n].(IdentityMatrix[n + 1] r^Range[0,n]).Inverse[CoeffMat[n]]

scal[r_][mat_] := scalmat[r, Length[mat] - 1].mat

Table[MatrixForm@scalmat[r, k], {k, 0, 5}]
```

$$\left\{(1), \begin{pmatrix} 1 & 0 \\ 0 & r \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & r & -\frac{r}{2}+\frac{r^2}{2} \\ 0 & 0 & r^2 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & r & -\frac{r}{2}+\frac{r^2}{2} & \frac{r}{3}-\frac{r^2}{2}+\frac{r^3}{6} \\ 0 & 0 & r^2 & -r^2+r^3 \\ 0 & 0 & 0 & r^3 \end{pmatrix},\right.$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & r & -\frac{r}{2}+\frac{r^2}{2} & \frac{r}{3}-\frac{r^2}{2}+\frac{r^3}{6} & -\frac{r}{4}+\frac{11\,r^2}{24}-\frac{r^3}{4}+\frac{r^4}{24} \\ 0 & 0 & r^2 & -r^2+r^3 & \frac{11\,r^2}{12}-\frac{3\,r^3}{2}+\frac{7\,r^4}{12} \\ 0 & 0 & 0 & r^3 & -\frac{3\,r^3}{2}+\frac{3\,r^4}{2} \\ 0 & 0 & 0 & 0 & r^4 \end{pmatrix},$$

$$\left.\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & r & -\frac{r}{2}+\frac{r^2}{2} & \frac{r}{3}-\frac{r^2}{2}+\frac{r^3}{6} & -\frac{r}{4}+\frac{11\,r^2}{24}-\frac{r^3}{4}+\frac{r^4}{24} & \frac{r}{5}-\frac{5\,r^2}{12}+\frac{7\,r^3}{24}-\frac{r^4}{12}+\frac{r^5}{120} \\ 0 & 0 & r^2 & -r^2+r^3 & \frac{11\,r^2}{12}-\frac{3\,r^3}{2}+\frac{7\,r^4}{12} & -\frac{5\,r^2}{6}+\frac{7\,r^3}{4}-\frac{7\,r^4}{6}+\frac{r^5}{4} \\ 0 & 0 & 0 & r^3 & -\frac{3\,r^3}{2}+\frac{3\,r^4}{2} & \frac{7\,r^3}{4}-3\,r^4+\frac{5\,r^5}{4} \\ 0 & 0 & 0 & 0 & r^4 & -2\,r^4+2\,r^5 \\ 0 & 0 & 0 & 0 & 0 & r^5 \end{pmatrix}\right\}$$

polynomial approximation of non-polynomial functions

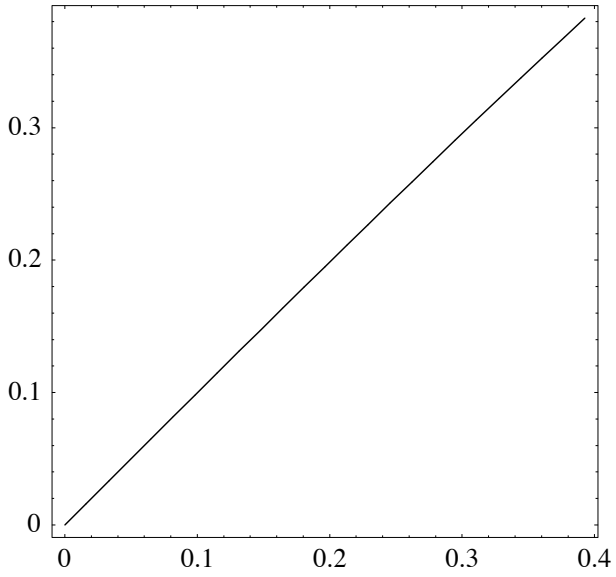
Now, the real point of the Difference Engine is not to tabulate polynomial functions for their own sake, but because they can be good approximations of non-polynomial functions, like logarithmic and trigonometric functions. There are several ways to come up with the coefficients to supply the Difference Engine, and they do not all have the same accuracy.

```
RMS[x_] := √Mean[x^2]

MaxAbs[x_] := Max@Abs@x

f[x_] := Sin[x]; xmin = 0; xmax = π / 8; ordermax = 7; deltax = (xmax - xmin)/ordermax; dx = (π / 180)/60;
```

```
Plot[f[x], {x, xmin, xmax},
  Frame → True,
  AspectRatio → Automatic
 ];
```




```
coarsedat = Table[{x, f[x]}, {x, xmin, xmax, (xmax - xmin)/ordermax}]
```

$$\begin{pmatrix} 0 & 0 \\ \frac{\pi}{56} & \text{Sin}[\frac{\pi}{56}] \\ \frac{\pi}{28} & \text{Sin}[\frac{\pi}{28}] \\ \frac{3\pi}{56} & \text{Sin}[\frac{3\pi}{56}] \\ \frac{\pi}{14} & \text{Sin}[\frac{\pi}{14}] \\ \frac{5\pi}{56} & \text{Sin}[\frac{5\pi}{56}] \\ \frac{3\pi}{28} & \text{Sin}[\frac{3\pi}{28}] \\ \frac{\pi}{8} & \text{Sin}[\frac{\pi}{8}] \end{pmatrix}$$

```
finedat = N[Table[{x, f[x]}, {x, xmin, xmax, dx}], 31];

finedat[[347]]
```

```
{0.10064732019833967226926616764 95, 0.10047748197919345217575038214 21}
```

```
taylorleft[x_] = Normal[Series[f[x], {x, xmin, ordermax}]]
```

$$x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040}$$

```
taylorleftdat = N[Table[{x, taylorleft[x]}, {x, xmin, xmax, dx}], 31];
```

```
taylormid[x_] = Normal[Series[f[x], {x, (xmin + xmax)/2, ordermax}]]
```

$$\left(-\frac{\pi}{16}+x\right)\text{Cos}\left[\frac{\pi}{16}\right]-\frac{1}{6}\left(-\frac{\pi}{16}+x\right)^3\text{Cos}\left[\frac{\pi}{16}\right]+\frac{1}{120}\left(-\frac{\pi}{16}+x\right)^5\text{Cos}\left[\frac{\pi}{16}\right]-\frac{\left(-\frac{\pi}{16}+x\right)^7\text{Cos}\left[\frac{\pi}{16}\right]}{5040}+$$
$$\text{Sin}\left[\frac{\pi}{16}\right]-\frac{1}{2}\left(-\frac{\pi}{16}+x\right)^2\text{Sin}\left[\frac{\pi}{16}\right]+\frac{1}{24}\left(-\frac{\pi}{16}+x\right)^4\text{Sin}\left[\frac{\pi}{16}\right]-\frac{1}{720}\left(-\frac{\pi}{16}+x\right)^6\text{Sin}\left[\frac{\pi}{16}\right]$$

```
taylormiddat = N[Table[{x, taylormid[x]}, {x, xmin, xmax, dx}], 31];

coe = DeCrankThrough[N[coarsedat[[All, 2]], 31]]
```

```
{0, 0.056070447237191788819071956605946,
 -0.000176418371075717917912733196846 9, -0.000175863293611802743902901162 2,
 1.10840844876997803508598 9×10^-6, 5.4984352180334886498906 4×10^-7,
 -5.21747419356410003203×10^-9, -1.71359503801127085965×10^-9}
```

```
coe2 = scal[dx/((xmax-xmin)/ordermax)][coe]
```

```
{0, 0.000290888204574310562161082705 6,
 -2.46140697046146961482402×10^-11, -2.46137750951349849484589 4×10^-11,
 4.11992750838553841261×10^-18, 2.083040749207098959401 1×10^-18,
 -2.008515134489175645×10^-24, -1.726863843775135025 99×10^-25}
```

```
sd = CrankTimes[coe2, (xmax - xmin)/dx];

scaldat = Transpose[{Table[x, {x, xmin, xmax, dx}], sd}];
```

```
General::spell1 :
 Possible spelling error: new symbol name "scaldat" is similar to existing symbol "scalmat". More…
```

```
lse[x_] =
 Fit[Table[{x, f[x]}, {x, xmin, xmax, (xmax - xmin)/ordermax}], Table[x^k, {k, 0, ordermax}], x]
```

```
-8.25994×10^-17 + 1. x - 1.73×10^-9 x^2 - 0.166667 x^3 -
 2.68698×10^-7 x^4 + 0.00833464 x^5 - 3.41725×10^-6 x^6 - 0.000194421 x^7
```

Note that Fit works only with machine precision. The more general function FindFit allows one to specify precision, though.

```
lse[x_] = Sum[a[k] x^k, {k, 0, ordermax}] /.
  FindFit[Table[{x, f[x]}, {x, xmin, xmax, (xmax - xmin)/ordermax}], Sum[a[k] x^k, {k, 0, ordermax}],
   Table[a[k], {k, 0, ordermax}], x, WorkingPrecision → 31]
```

```
-3.3078643043606381451452978956 47×10^-34 + 1.0000000000379235260629414035 35 x -
 1.7288037798952994411314648044 82×10^-9 x^2 - 0.1666666363779967473501317126 377 x^3 -
 2.6862188976022754431768391953 15×10^-7 x^4 + 0.0083346397056521099020258147 62716 x^5 -
 3.4168817924261343196603526857 87×10^-6 x^6 - 0.0001944217015353643214087538209901 x^7
```

```
lsedat = N[Table[{x, lse[x]}, {x, xmin, xmax, dx}], 31];

namelist = {"taylorleft", "taylormid", "lse", "scal"};
```

```
TableForm[err = {RMS[(# - finedat)[[All, 2]]], MaxAbs[(# - finedat)[[All, 2]]]} & /@
   ToExpression[# <> "dat" & /@ namelist],
 TableHeadings → {namelist, {"RMS error", "max abs error"}}
]
```

	RMS error	max abs error
taylorleft	$1.406738326095488426143 \times 10^{-10}$	$6.111773653887720340837 \times 10^{-10}$
taylormid	$2.6209508780388809061 \times 10^{-12}$	$1.18568470628245997265 \times 10^{-11}$
lse	$1.125076552427438991 \times 10^{-13}$	$3.34123032623967236 \times 10^{-13}$
scal	$1.1250765524274 \times 10^{-13}$	$3.3412303262397 \times 10^{-13}$

```
-Log[10, err]
```

$$\begin{pmatrix} 9.851786680178482304311 & 9.213832737987307522139 5 \\ 11.58154111854083513669 & 10.926030781904423044 16 \\ 12.948817926294204846 & 12.476093585431277196 \\ 12.94881792629420 & 12.47609358543128 \end{pmatrix}$$

Our tables should be good for about this many digits:

```
maxdig = Ceiling[-Log[10, err[[3, 1]]]]
```

```
13
```

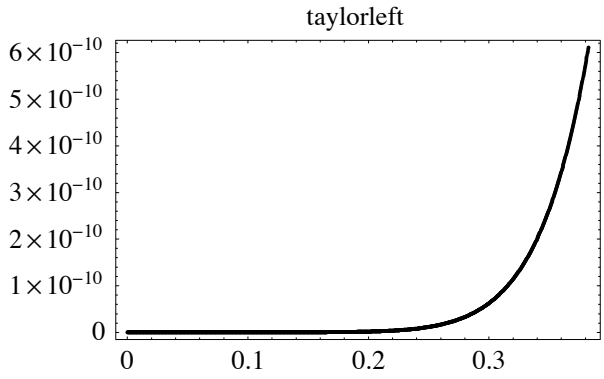
```
Most@# / Rest@# &@err
```

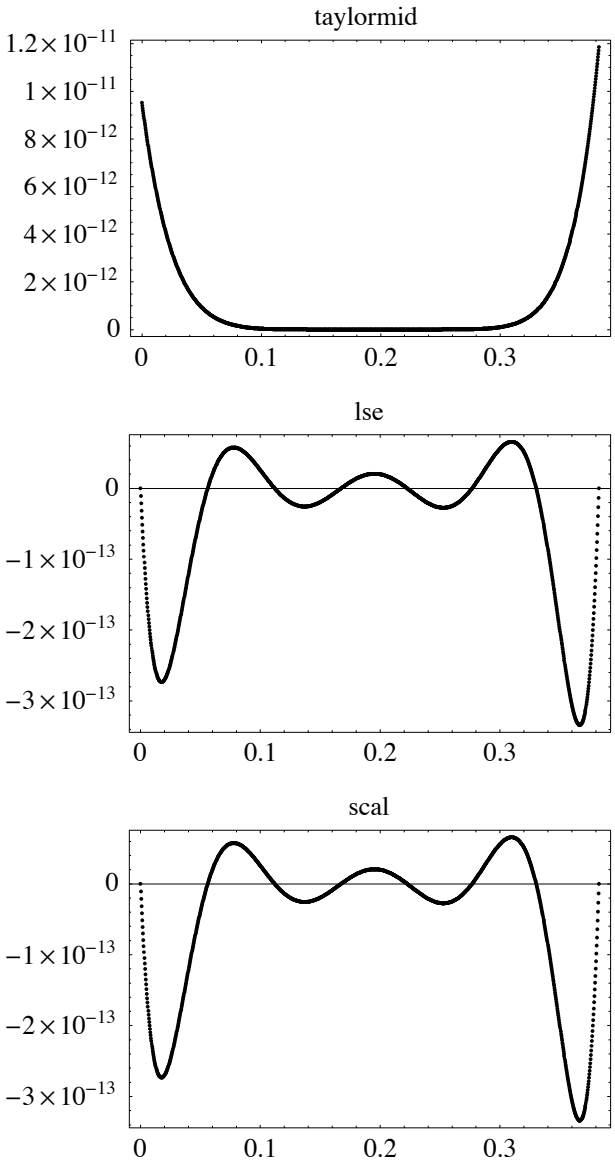
$$\begin{pmatrix} 53.672823015594874732 & 51.546364910535852510 \\ 23.29575594108666008 & 35.4864702672822775 \\ 1.0000000000000 & 1.0000000000000 \end{pmatrix}$$

We see that the rms error and maximum error are quite poor for the Taylor series starting from one side, somewhat better for the Taylor series centered about the midpoint of the interval, and quite good for the methods of scaled coefficients or least squared error minimization.

```
ysub[a_, b_] := Transpose@{a[[All, 2]], a[[All, 2]] - b[[All, 2]]}

ListPlot[ysub[finedat, ToExpression[# <> "dat"]],
    PlotLabel → #,
    Frame → True,
    PlotRange → All
   ] & /@ namelist;
```





We can easily export the table of values as a text file:

```
Spaces[n_] := StringJoin[Table[" ", {n}]]
```

```
Export["BabbageSin225.txt",
 {#1, #2, #3, #4, #5} & @@@
  ({{"  d", "  m", " dd", Spaces[maxdig + 4] <> "sin(x)", Spaces[maxdig + 3] <> "error"}} ~
    Join ~ ({ToString@NumberForm[IntegerPart[#1 180/π], 2, NumberPadding → {" ", "0"}],
        ToString@NumberForm[60 FractionalPart[#1 180/π], 2, NumberPadding → {" ", "0"}],
        ToString@NumberForm[N[#1, maxdig], {maxdig, maxdig},
          NumberPadding → {" ", "0"}],
        ToString@NumberForm[#2, {maxdig, maxdig}, NumberPadding → {" ", "0"}],
        ToString@NumberForm[FortranForm[#2 - f[#1]], 3]} & @@@ scaldat[[All, All]])),
 "TSV"
]
```

```
BabbageSin225.txt
```

Appendix

Taylor series

One of the useful results of calculus is that a function may be approximated to any desired accuracy near a point as a power series whose coefficients are determined by the value of the function and its derivatives at that point. This is related to the way the Difference Engine works.

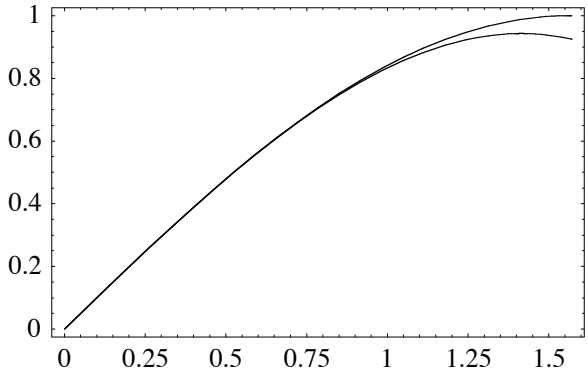
```
Series[g[x], {x, x0, 7}]
```

$$g[x0] + g'[x0]\,(x - x0) + \frac{1}{2} g''[x0]\,(x - x0)^2 + \frac{1}{6} g^{(3)}[x0]\,(x - x0)^3 + \frac{1}{24} g^{(4)}[x0]\,(x - x0)^4 + \frac{1}{120} g^{(5)}[x0]\,(x - x0)^5 + \frac{1}{720} g^{(6)}[x0]\,(x - x0)^6 + \frac{g^{(7)}[x0]\,(x - x0)^7}{5040} + O[x - x0]^8$$

```
Series[Sin[x], {x, 0, 7}]
```

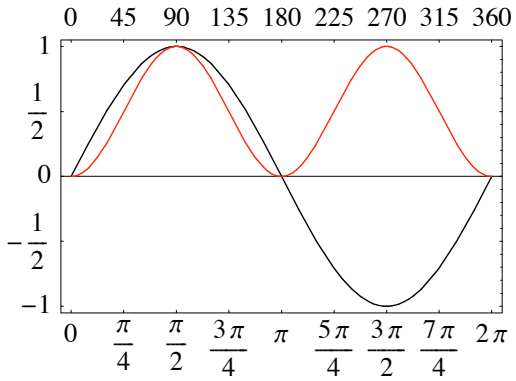
$$x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + O[x]^8$$

```
Plot[Evaluate@{Sin[x], Normal@Series[Sin[x], {x, 0, 3}]},
  {x, 0, π / 2},
  Frame → True
 ];
```



Trigonometric functions

```
Plot[{Sin[θ], Sin[θ]^2}, {θ, 0, 2 π},
  Frame → True,
  FrameTicks → {Range[0, 2 π, π / 4],
    Range[-1, 1, 1 / 2],
    {#, 180/π #} & /@ Range[0, 2 π, π / 4],
    Automatic},
  PlotStyle → {GrayLevel[0], Hue[0]}
 ];
```



The trigonometric functions are so related to each other that we need only square roots and the elementary arithmetic operations to convert one to another, so we will choose sin.

Since the trig functions are functions on a circular domain, we only have to compute them for $0 \le \theta \le 2\pi$ or $-\pi \le \theta \le \pi$, etc. Since $\sin(\pi - \theta) = -\sin(\theta)$, we only need to consider $0 \le \theta \le \pi$. Then, $\sin(\pi/2 - \theta) = \sin(\pi/2 + \theta)$, so we can further reduce the domain to $0 \le \theta \le \pi/2$. In fact, we can repeat this halving as many times as we wish, since $\sin^2(\theta) = \frac{1+\sin(2\theta)}{2}$, so we could fold to get $0 \le \theta \le \pi/4$, etc.

Deriving the coefficient matrix by direct brute force:

```
CoeffMat[omax_] := Module[{dat, coeff},
  dat = Table[
    Normal[Series[a[x], {x, 0, omax}]] /. x → k dx,
    {k, 0, omax}];
  coeff = DeCrankThrough[dat];
  PadRight[#, omax + 1] & /@
   CoefficientList[coeff /. {a[0] → 1, Derivative[n_][a][0] → dxa^n dx^-n n!}, dxa]
 ]
```

Table[MatrixForm@CoeffMat[k],{k,8}]

Fixed-point real numbers and ten's complement notation

The Difference Engine does not use floating-point arithmetic, but rather fixed-point. What is stored in each register is a 31-digit decimal integer. The operator must decide how to scale the coefficents so that overflow and underflow are avoided and as little precision as possible is lost. In effect, the operator decides where to put the decimal point, but the decimal point does not not exist in the machinery.

Humans typically represent negative numbers by a negative sign followed by the absolute value, such as -736.23, but it is much simpler to build a machine to work by storing negative numbers as ten's complement (or in the case of modern binary machines, two's complement). Each digit is subtracted from one less than the base, then a one is added to the smallest digit. The result is a number padded on the right by zeroes and on the left by zeros (for a positive number) or nines (for a negative number). Then the sum of two numbers can be computed using the same machinery regardless of the signs of the numbers.

```
rep[x_] := IntegerDigits[x 1*^25, 10, 31]

neg[digs_] := IntegerDigits[FromDigits[9 - digs] + 1, 10, 31]

add[a_, b_] := IntegerDigits[FromDigits[a] + FromDigits[b], 10, 31]

rep[3]
```

```
{0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
neg@rep[3]
```

```
{9, 9, 9, 9, 9, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
rep[5]
```

```
{0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
add[neg@rep@3, rep@5]
```

```
{0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
rep[782 + 49/100]
```

```
{0, 0, 0, 7, 8, 2, 4, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
neg@rep[782 + 49/100]
```

```
{9, 9, 9, 2, 1, 7, 5, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
rep[982 + 49/100]
```

```
{0, 0, 0, 9, 8, 2, 4, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
add[neg@rep[741 + 23/100], rep[982 + 49/100]]
```

```
{0, 0, 0, 2, 4, 1, 2, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

Original version

endDegree = 360/16

starting degree assumed to be 0

make computing polynomial, assumed to be 8th degree x^7

```
Table[{i, N[i π/56], N@Sin[i π/56]}, {i, 0, 7}]
```

```
arrayValues = Table[N[Sin[i π/56], 31], {i, 0, 7}]
```

arrayPoly = Table[x^i, {i, 0, 8}]

original polynomial

arrayFitPoly = Fit[arrayValues,arrayPoly,x]

arrayStepValues = Table[N[Sin[i/60*Pi/180],31], {i, 0, 7}]

(* arrayStepValues = {0, 1, 8, 27, 64, 125, 216, 343 } *)

```
arrayStepValues 7/arrayStepValues[[-1]]
```

```
For [i=1, i<9, i++,
  {  Print [i," ",N[i*cycleStep*PI/180]," ",N[arrayStepValues[[i]] ,40] ] ;
    Write[stmp, i ,"  " ,arrayStepValues[[i]] ]
  }]
```

1 0.00555556 cycleStep PI 0

2 0.0111111 cycleStep PI 0.00029088820456342459637429741574 00

3 0.0166667 cycleStep PI 0.00058177638451306761061439521430 67

4 0.0222222 cycleStep PI 0.00087266451523514954330458929907 38

5 0.0277778 cycleStep PI 0.00116355257211589506046599038330 6

6 0.0333333 cycleStep PI 0.00145444053054153507627449081711 1

7 0.0388889 cycleStep PI 0.00174532836589830883577820272085 0

8 0.0444444 cycleStep PI 0.00203621605357246599761419119387 3

```
(* initialize and do difference array *)
arrayDifferences = Table[N[0,31], {i, 1, 8 }]

 (* initialize *)
For [ i=1, i<=8, i++,
  {
   arrayDifferences[[i]] = arrayStepValues[[1]] ;
   For [ ii=1, ii<=8-i, ii++,
      {arrayStepValues[[ii]]=arrayStepValues[[ii+1]]-arrayStepValues[[ii]]
      }]
   }]

?arrayDifferences

Global`arrayDifferences

arrayDifferences =
 {0, 0.00029088820456342459637429741574 00, -2.4613781582134199617173 × 10^-11,
  -2.4613779499415704096626 × 10^-11, 4.165436814809891 × 10^-18,
  2.08271796682695 × 10^-18, -5.2869355 × 10^-25, -1.762311 × 10^-25}

let's crank the Babbage machine a few times

degrees = 0;

Print ["endDegree = ", endDegree] ;

endDegree = 22.5

For [ n=0, n<1500, n++,
  {
   minutes = Mod[{n * minutesStep}, 60] ;
   degrees = Floor[{n * minutesStep / 60} ] ;
  (* Print [      degrees, " ",minutes, " ", arrayDifferences[[1]] ] ; *)
```

```
    Write [stmp, degrees, " ",minutes, " ", arrayDifferences[[1]] ] ;
    (* do reference value *)
    reference = N[ Sin[n*minutesStep/60*Pi/180] , 31] ;
    (*Print ["   reference = ", reference ]; *)
    If [ reference != 0,
      {
       error = reference - arrayDifferences[[1]] ;
       percentError = N[error/reference*100, 31] ;
       (* Print [      "       ", percentError, " %" ] ; *)
        Write [stmp, "       ", percentError, " %"  ] ;
      }] ;
    For [ i=1, i<8, i++,
     {arrayDifferences[[i]]=arrayDifferences[[i]]+arrayDifferences[[i+1]] }
      ] ;
   }] ;

Close [stmp]

Print ["done"]

done
```

Original text

http://www.ed-thelen.org/bab/MA-S-23code.html

```
(* Print ["Good Morning Mr. Battage, hope you slept well."] *)

stmp = OpenWrite["MA-s-23.txt"]
Write[stmp, Good Morning Mr. OrganGrinder]
Write[stmp, Please note - this example is _not_ pipelined]

endDegree = 22.5  (* starting degree assumed to be 0  *)
(* make computing polynomial, assumed to be 8th degree x^7 *)
x=. (* clear any previous value *)
arrayValues = Table[N[Sin[i*endDegree/7*Pi/180],31], {i,0,7}]
?arrayValues
Print ["printing ?arrayValues"]
Write[stmp, "writing ?arrayValues"]
For [i=1, i<9, i++,
  {  Print [i,"  ", N[i*endDegree/7*PI/180], "  ", N[arrayValues[[i]] ,31] ] ;
    Write[stmp, i ,"  " ,arrayValues[[i]] ]
   }]
arrayPoly = Table[x^i, {i, 0, 8}]
?arrayPoly
arrayFitPoly = Fit[arrayValues,arrayPoly,x]  (* orig polynomial *)
?arrayFitPoly
```

```
minutesStep = 1 (* minutes *)
arrayStepValues = Table[ N[Sin[i*minutesStep/60*Pi/180],31], {i, 0, 7}]
(*  arrayStepValues = {0, 1, 8, 27, 64, 125, 216, 343 }  *)
Print ["printing arrayStepValues"]
Write[stmp, "writing ?arrayStepValues"]
For [i=1, i<9, i++,
  {  Print [i," ",N[i*cycleStep*PI/180]," ",N[arrayStepValues[[i]] ,40] ] ;
    Write[stmp, i ,"  " ,arrayStepValues[[i]] ]
  }]

(* initialize and do difference array *)
arrayDifferences = Table[N[0,31], {i, 1, 8 }]  (* initialize *)
For [ i=1, i<=8, i++,
  {
   arrayDifferences[[i]] = arrayStepValues[[1]] ;
   For [ ii=1, ii<=8-i, ii++,
      {arrayStepValues[[ii]]=arrayStepValues[[ii+1]]-arrayStepValues[[ii]]
      }]
  }]
?arrayDifferences

(* lets crank the Babbage machine a few times *)
stringText = "lets crank the Babbage machine a few times"
Print [stringText] ;
Write [stmp, stringText] ;
degrees = 0;
Print ["endDegree = ", endDegree] ;
For [ n=0, n<1500, n++,
  {
   minutes = Mod[{n * minutesStep}, 60] ;
   degrees = Floor[{n * minutesStep / 60} ] ;
  (* Print [      degrees, " ",minutes, " ", arrayDifferences[[1]] ] ; *)
   Write [stmp, degrees, " ",minutes, " ", arrayDifferences[[1]] ] ;
   (* do reference value *)
   reference = N[ Sin[n*minutesStep/60*Pi/180] , 31] ;
   (*Print ["   reference = ", reference ]; *)
   If [ reference != 0,
     {
      error = reference - arrayDifferences[[1]] ;
      percentError = N[error/reference*100, 31] ;
      (* Print [      "       ", percentError, " %" ] ; *)
       Write [stmp, "       ", percentError, " %"  ] ;
     }] ;
   For [ i=1, i<8, i++,
    {arrayDifferences[[i]]=arrayDifferences[[i]]+arrayDifferences[[i+1]] }
     ] ;
  }] ;
```

```
Close [stmp]
Print ["done"]
```