Polynomial Solver Algorithm for the TI Graphing Calculators

Michael Lloyd, Ph.D.
Professor of Mathematics

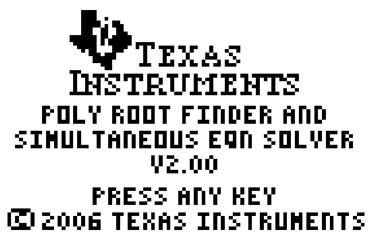
Abstract

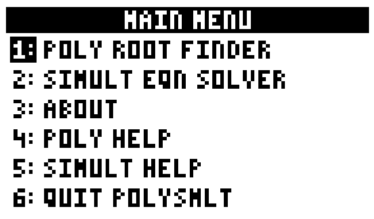
The history of polynomial solvers for the TI calculators (TI-85 through the TI-nspire), and an explanation of a QR-based polynomial solver algorithm is explained.

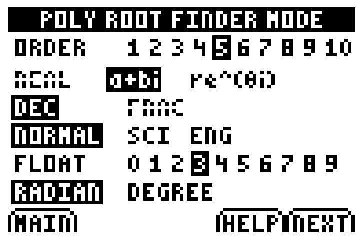
TIs with Built-In Polynomial Solvers

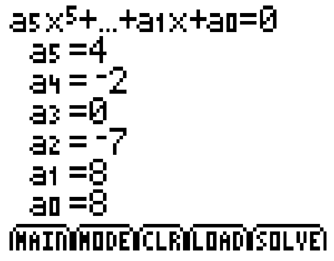
The TI-83+ appeared in 1999, and the original TI-83+ polynomial application appeared in 2001. This application also runs on a TI-84+ and the screen shots for a newer version of this software as used to find the zeros of $4x^5 - 2x^4 - 7x^3 + 8x + 8$ are shown here:

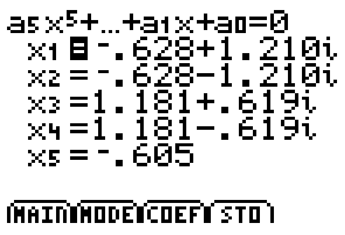
TI-84+ Silver Edition


```
TEXAS
INSTRUMENTS
POLY ROOT FINDER AND
SIMULTANEOUS EQN SOLVER
V2.00
PRESS ANY KEY
© 2006 TEXAS INSTRUMENTS
```


```
MAIN MENU
1: POLY ROOT FINDER
2: SIMULT EQN SOLVER
3: ABOUT
4: POLY HELP
5: SIMULT HELP
6: QUIT POLYSMLT
```

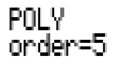

```
POLY ROOT FINDER MODE
ORDER   1 2 3 4 [✓ 5] 6 7 8 9 10
REAL  [✓ a+bi]  re^(θi)
[✓ DEC]  FRAC
[✓ NORMAL]  SCI  ENG
FLOAT   0 1 2 [✓ 3] 4 5 6 7 8 9
[✓ RADIAN]  DEGREE
MAIN                 HELP NEXT
```


```
a5x^5+...+a1x+a0=0
 a5 =4
 a4 =-2
 a3 =0
 a2 =-7
 a1 =8
 a0 =8
MAIN MODE CLR LOAD SOLVE
```

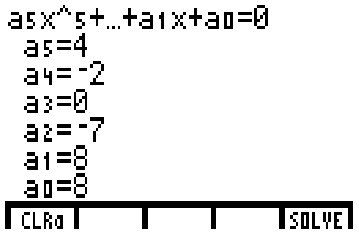

```
a5x^5+...+a1x+a0=0
 x1 =-.628+1.210i
 x2 =-.628-1.210i
 x3 =1.181+.619i
 x4 =1.181-.619i
 x5 =-.605

MAIN MODE COEF STO
```

The TI-85 and TI-86 appeared in 1992 and 1997, respectively, and both of these calculators are officially discontinued. The screen shots shown here are from its built-in polynomial solver menu used to solve the above polynomial. Note that complex numbers on the TI-85 and TI-86 are displayed as ordered pairs.

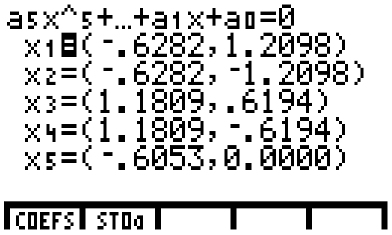
TI-86



```
POLY
order=5
```



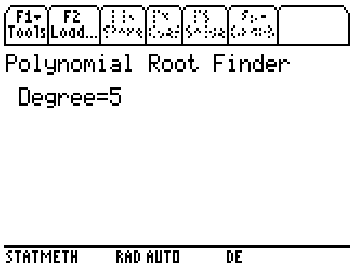
```
a5x^5+…+a1x+a0=0
 a5=4
 a4=-2
 a3=0
 a2=-7
 a1=8
 a0=8
CLRa                SOLVE
```



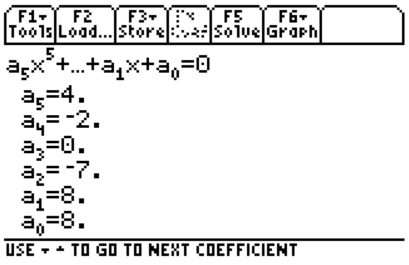
```
a5x^5+…+a1x+a0=0
 x1=(-.6282,1.2098)
 x2=(-.6282,-1.2098)
 x3=(1.1809,.6194)
 x4=(1.1809,-.6194)
 x5=(-.6053,0.0000)
COEFS STOa
```

The 89 appeared in 1998 and although it was a computer-algebra system (CAS), a polynomial application for it appeared in 2001, the same year the polynomial application was released for the TI-83+.

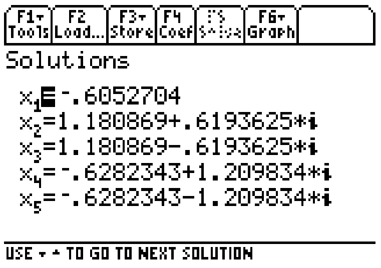
TI-89 Titanium



```
F1 Tools F2 Load...
Polynomial Root Finder
 Degree=5
STATMETH    RAD AUTO    DE
```



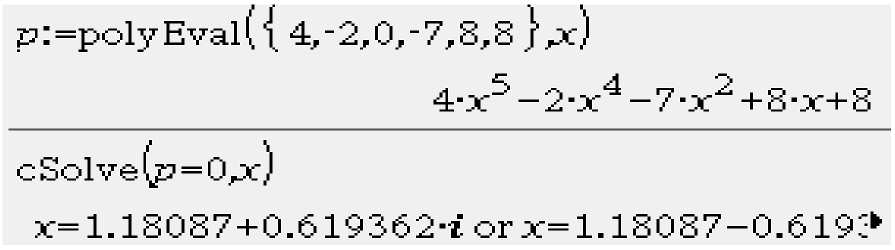
```
F1 Tools F2 Load... F3 Store F5 Solve F6 Graph
a5x^5+…+a1x+a0=0
 a5=4.
 a4=-2.
 a3=0.
 a2=-7.
 a1=8.
 a0=8.
USE ↓ ↑ TO GO TO NEXT COEFFICIENT
```



```
F1 Tools F2 Load... F3 Store F4 Coef F6 Graph
Solutions
 x1=-.6052704
 x2=1.180869+.6193625*i
 x3=1.180869-.6193625*i
 x4=-.6282343+1.209834*i
 x5=-.6282343-1.209834*i
USE ↓ ↑ TO GO TO NEXT SOLUTION
```

The TI non-CAS and CAS nspires appeared in 2007, and the touch pad nspires appeared in 2010. The screen shot here shows finding the zeros using a CAS nspire.

TI-Touch Pad Nspire

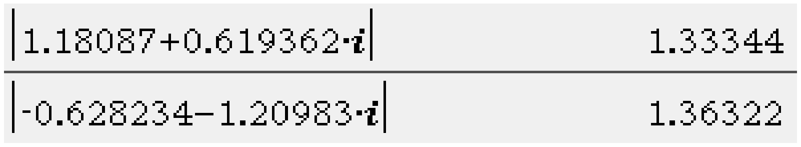


```
p:=polyEval({4,-2,0,-7,8,8},x)
                    4·x^5-2·x^4-7·x^2+8·x+8
cSolve(p=0,x)
 x=1.18087+0.619362·i or x=1.18087-0.6193▸
```

Basic Polynomial Solver Programs

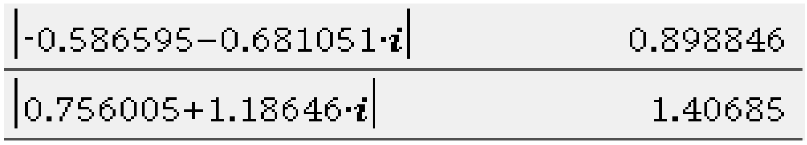
The TI-83 appeared in 1996 and the first version of the polynomial solver program was written for TI-83 and TI-82 by the author the same year. This program hangs indefinitely if any two pairs of complex zeros have similar moduli. For example, the above polynomial has two such pairs of complex zeros causing the program to run indefinitely without successfully finding all the zeros.

Here are the two zeros with similar moduli that foiled the program.



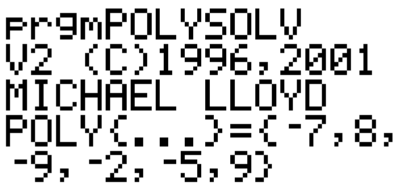
```
|1.18087+0.619362·i|          1.33344
|-0.628234-1.20983·i|         1.36322
```

The program runs successfully for the polynomial $-7x^5 + 8x^4 - 9x^3 - 2x^2 - 5x + 9$ because the two pair of complex zeros have sufficiently different moduli:

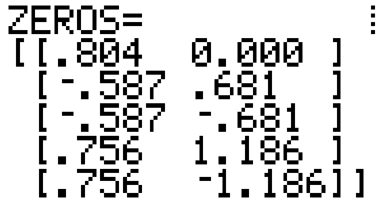


```
|-0.586595-0.681051·i|        0.898846
|0.756005+1.18646·i|          1.40685
```

TI-83



```
prgmPOLYSOLV
V2 (C)1996,2001
MICHAEL LLOYD
POLY(...)={-7,8,
 -9,-2,-5,9}
```



```
ZEROS=
[[.804  0.000 ]
 [-.587 .681  ]
 [-.587 -.681 ]
 [.756  1.186 ]
 [.756  -1.186]]
```

Each row in the output matrix gives a complex zero.

The following table gives a timeline for the TI graphing calculators and the various polynomial solvers.

A = flash application, B = basic program or function, C = built-in command

Year	**90**	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**00**	**01**
TI	81		85	82		80,92	83	86	73,89	92+,83+		83+SE
solver			C			C	B-82,83	C	C	C		A83+,89

Year	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**

TI	V200		84+,89T			nspires			Nspire touch
solver	C		C-89T			C-nspire		B-nspire	

Basic Polynomial Method using Eigenvalues

The non-CAS TI-nspire does not have a polynomial command, but it does have a built-in command for finding all the eigenvalues of a matrix. This suggests the following method:

1. Divide the polynomial by its leading coefficient.
2. Create an upper Hessenburg matrix whose eigenvalues are the same as the zeros of the polynomial.
3. Use the eigenvalue command.

Recall that a Hessenburg matrix is almost triangular. Specifically, an upper Hessenburg matrix has zeros below the subdiagonal. The polynomial $p = x^n - a_1 x^{n-1} - a_2 x^{n-2} - \cdots - a_{n-1}x - a_n$ has the same eigenvalues as the upper-Hessenburg matrix *A* shown here.

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_n \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}_{n\times n}$$

To prove this equivalence, use the definition of the characteristic polynomial and expand along the first column:

$$|xI_n - A| = \begin{vmatrix} x-a_1 & -a_2 & -a_3 & \cdots & -a_n \\ -1 & x & 0 & \cdots & 0 \\ 0 & -1 & x & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & x \end{vmatrix}_{n\times n}$$

$$= (x-a_1)\begin{vmatrix} x & 0 & 0 & \cdots & 0 \\ -1 & x & 0 & \cdots & 0 \\ 0 & -1 & x & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & x \end{vmatrix}_{(n-1)\times(n-1)} + \begin{vmatrix} -a_2 & -a_3 & -a_4 & \cdots & -a_n \\ -1 & x & 0 & \cdots & 0 \\ 0 & -1 & x & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & x \end{vmatrix}_{(n-1)\times(n-1)}$$

$$= (x-a_1)x^{n-1} - a_2\begin{vmatrix} x & 0 & 0 & \cdots & 0 \\ -1 & x & 0 & \cdots & 0 \\ 0 & -1 & x & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & x \end{vmatrix}_{(n-2)\times(n-2)} + \begin{vmatrix} -a_3 & -a_4 & -a_5 & \cdots & -a_n \\ -1 & x & 0 & \cdots & 0 \\ 0 & -1 & x & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & x \end{vmatrix}_{(n-2)\times(n-2)}$$

$$= x^n - a_1x^{n-1} - a_2x^{n-2} - a_3\begin{vmatrix} x & 0 & 0 & \cdots & 0 \\ -1 & x & 0 & \cdots & 0 \\ 0 & -1 & x & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & x \end{vmatrix}_{(n-3)\times(n-3)} + \begin{vmatrix} -a_3 & -a_4 & -a_5 & \cdots & -a_n \\ -1 & x & 0 & \cdots & 0 \\ 0 & -1 & x & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & x \end{vmatrix}_{(n-3)\times(n-3)}$$

$$= \cdots = x^n - a_1x^{n-1} - a_2x^{n-2} - \cdots - a_{n-2}\begin{vmatrix} x & 0 \\ -1 & x \end{vmatrix} + \begin{vmatrix} -a_{n-1} & -a_n \\ -1 & x \end{vmatrix}$$

$$= x^n - a_1x^{n-1} - a_2x^{n-2} - \cdots - a_{n-2}x^2 - a_{n-1}x - a_n = p$$

Detailed Polynomial Solver Method based on a QR-Algorithm

Our algorithm will depend upon the Real Schur Decomposition of a square real matrix:

- If A is a real n×n matrix, there exists an orthogonal matrix Q such that $Q^TAQ = R$ is quasi-upper triangular.
- A quasi-upper triangular matrix is an upper-Hessenburg matrix whose eigenvalues can be obtained from the 1×1 and 2×2 blocks along its diagonal.

This is analogous to the factorization of a real polynomial into linear and irreducible quadratic factors.

A Given's Matrix is defined to be an n×n matrix G = J(I,j,θ) where $1 \le I < j \le n$

- $g_{kk} = 1$ if $i \ne k$ and $j \ne k$
- $g_{ii} = g_{jj} = \cos\theta$
- $g_{ij} = \sin\theta$
- $g_{ji} = -\sin\theta$
- $g_{kl} = 0$, otherwise
- Given's matrices are orthogonal.
- They are basically an identity matrices embedded with a rotation in the i-j plane.

$$\begin{bmatrix} 1\ddots & 0 & \cdots & 0 & 0 \\ 0 & \cos\theta & 0 & \sin\theta & 0 \\ \vdots & 0 & \ddots 1 \ddots & 0 & \vdots \\ 0 & -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & \cdots & 0 & \ddots 1 \end{bmatrix}$$

The following simplified Q-R Algorithm was used, but the deflation was actually done in the lower right corner in the TI-83 and TI-82 programs.

1. Start with upper-Hessenburg matrix H.
2. Set all subdiagonals to zero that satisfy $|h_{i,i-1}| \le \varepsilon(|h_{ii}| + |h_{i-1,i-1}|)$.
3. Apply Hessenburg Q-R step.
   a) Obtain R = QH using Given's matrices.
   b) Replace H with RQ^T.
4. Deflate (break H apart where 0 appears in lower diagonal).
5. Stop when H is 1×1 or 2×2.

Here is the Q-R Part of TI-83 program:

Create Given's Matrices	**QH Step**	**RQ^T Step**
For(K,1,N-1) [A](K,K)\->\A [A](K+1,K)\->\B If B=0 Then 1\->\C:0\->\S Else If abs(B)\>=\abs(A) Then A/B\->\T 1/\root\(1+T\^2\)\->\S ST\->\C Else B/A\->\T 1/\root\(1+T\^2\)\->\C CT\->\S End:End	C\->\\L\TEMPC(K) S\->\\L\TEMPS(K) For(J,K,N) C[A](K,J)+S[A](K+1,J)\->\A C[A](K+1,J)-S[A](K,J)\->\B A\->\[A](K,J) B\->\[A](K+1,J) End:End	For(K,1,N-1) \L\TEMPC(K)\->\C \L\TEMPS(K)\->\S For(I,1,K+1) C[A](I,K)+S[A](I,K+1)\->\A C[A](I,K+1)-S[A](I,K)\->\B A\->\[A](I,K) B\->\[A](I,K+1) End:End

The TI-82 and TI-83 programs use a crude upperbound for the moduli of all the polynomial zeros that the author found in a precalculus text:

Let $p = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0$

and $M = \max\limits_{k=0}^{n-1} |b_k| / |b_n| + 1$.

If z is any zero of p, then $|z| < M$.

Proof : If $|z| \le 1$, then clearly $|z| < M$.

So, we may assume that $|z| > 1$.

Let $N = \max\limits_{k=0}^{n-1} |b_k|$.

$$b_n z^n + b_{n-1} z^{n-1} + \cdots + b_1 z + b_0 = 0$$

$$|b_n||z|^n \le \sum_{k=0}^{n-1} |b_k||z|^k \le N \frac{|z|^n - 1}{|z| - 1} < N \frac{|z|^n}{|z| - 1}$$

$$|z|^{n+1} - |z|^n < \frac{N}{|b_n|} |z|^n$$

$$|z| < M + 1$$

The zeros approximated by the eigenvalues of the upper Hessenberg matrix were actually used as a starting point for a modified Newton's method as described below:

- This will converge quadratically even if the multiplicity of a zero is more than one.
- If p is a polynomial, substitute $f=p/p'$ in Newton's method to obtain the following iterative formula:

$$x_{n+1} = x_n - \frac{f}{f'}$$

$$x_{n+1} = x_n - \frac{pp'}{(p')^2 - pp''}$$

Recall that a sequence x_n is said to converge quadratically to c if there exists a positive, real constant K such that

$$\limsup_{n \to \infty} \frac{|x_{n+1} - c|}{|x_n - c|^2} = K$$

Here is the algorithm used to for the TI-82/83 programs:

1. Check if polynomial is linear, quadratic, or x^n-k=0. If yes, solve and stop. If not, continue to Step 2.
2. Create upper Hessenberg matrix.
3. Compute tolerance based on maximum possible zero.
4. Perform QR iteration until tolerance is met.
5. Break off 1×1 or 2×2 block.
6. Find approximate eigenvalue.
7. Find accurate eigenvalue using a modified Newton's method.
8. Perform synthetic division and replace polynomial with quotient to reduce its degree by 1 or 2.
9. Go to step 1.

References

- http://www.ticalc.org/basics/calculators/
- Graduate Numerical analysis notes (fall 1986)

Biographical Sketch

Michael Lloyd received his B.S in Chemical Engineering in 1984 and accepted a position at Henderson State University in 1993 shortly after earning his Ph.D. in Mathematics from Kansas State University. He has presented papers at meetings of the Academy of Economics and Finance, the American Mathematical Society, the Arkansas Conference on Teaching, the Mathematical Association of America, and the Southwest Arkansas Council of Teachers of Mathematics. He has also been an AP statistics consultant since 2002.

The Dictionary as Southern Cocktail: A Conversation with Roy Blount Jr.

Michael Ray Taylor, M.F.A.
Professor of Mass Media Communication

Abstract

An interview conducted via email with Southern writer and public radio humorist Roy Blount Jr.

Roy Blount Jr. is one of those rare writers whose actual voice has become almost as familiar as his literary one. Most weekends, you can hear his signature blend of Georgia drawl and rapid-fire wit on the NPR news quiz “Wait, Wait, Don’t Tell Me.” He periodically recites comical poetry and inflicts musical screeching (as founder of the fictional “Society for the Singing Impaired”) on Garrison Keillor’s “A Prairie Home Companion.” The Vanderbilt graduate has performed a successful off-Broadway one-man-show, appeared on several network TV shows, and stays busy on the college lecture circuit.

Despite all this talking, Blount has somehow managed to sit down and write 21 books, in a literary voice that, as Michael Dirda recently wrote in the Washington Post, “neatly balances real learning with easy-loping charm.” His penchant for pithy puns, political petards and periodic alliterative passages produces sentences that pull chuckles from readers; he piles these into paragraphs and punch lines that can make them positively puncture a gut. With such a facility for wordplay, it is no wonder that one of Blount’s many side gigs is serving as a usage adviser to *The American Heritage Dictionary*. Evidently unsatisfied with merely providing guidance to the dictionary, he has now written one of his own, *Alphabet Juice: The Energies, Gists, and Spirits of Letters, Words, and Combinations Thereof; Their Roots, Bones, Innards, Piths, Pips, and Secret Parts, ... With Examples of Their Usage Foul and Savory*, recently released in paper by Farrar, Strauss and Giroux. The book explores the provenance of an eclectic variety of words in short, invariably funny essays that average about a page or two long.

Writing for Chapter 16, an online publication of Humanities Tennessee, Michael Ray Taylor managed to catch up with Blount between radio appearances and deadlines for a conversation on the book, the South and the state of American letters.