Beyond Newton: Robust Methods for Solving Large Nonlinear Models in TROLL

Peter Hollinger
Intex Solutions, Inc.
peterh@intex.com

July 1, 2000

Abstract

Newton's method is an important algorithm for solving nonlinear systems of equations. For any solution algorithm, the principle concerns are robustness (finding a solution reliably) and efficiency (finding a solution quickly). Newton is simple in principle, but a useful implementation must deal with a variety of practical and theoretical obstacles.

By using partial derivatives, Newton's method can model the shape of the residual surface to provide quadratic convergence near the solution: the number of correct digits doubles each iteration. But the full step may be illegal, leading to economic nonsense like negative prices and numerical problems like taking the log of a negative number. Automatic backtracking — taking shorter steps along the Newton direction — can improve global convergence in such cases.

This paper describes enhancements to Newton's method used in the TROLL modeling system and illustrates them with a variety of contemporary models.

Acknowledgements

I would like to thank Hope Pioro of the Bank of Canada, who provided the stochastic simulation program and model that I used for some of the experiments. I would also like to thank Sarma Jayanthi and Doug Laxton of the International Monetary Fund for providing other test models. All errors are mine.

1. The problem

Given a system of equations:

$$F(x) = 0 \tag{1}$$

where F is a vector of equations and x is a vector of variables

and: $x^{(0)}$, an initial guess for x,

find a "solution" x^* such that $F(x^*) \approx 0$.

F is assumed to be differentiable, so the Jacobian matrix, $J \equiv \partial F / \partial x$ can be calculated.

For a macroeconometric model, the variables in x are typically timeseries and may appear in the model with lags or leads. In a backward-looking model with no leads, x would contain just the contemporaneous variables; the system can be solved period-by-period, with the solutions from one period supplying lagged values for later periods. In a perfect-foresight forward-looking model, with leads as well as lags, F and x consist of the original system stacked for multiple time periods over a predetermined horizon.

In either type of model, the Jacobian matrix J is typically very sparse. It can usually be permuted into a block-triangular form that allows the full system to be decomposed into smaller blocks of simultaneous equations that can be solved serially. In the case of a forward-looking model stacked over time, the Jacobian also has a repetitive block-band-diagonal structure.

2. Newton's method

Given a current candidate solution $x^{(k)}$, Newton's method calculates the Newton step:

$$\Delta x^{(k)} = -J(x^{(k)})^{-1} F(x^{(k)})$$

and generates a new iteration value x_{k+1}:

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)} = x^{(k)} - J(x^{(k)})^{-1} F(x^{(k)})$$

Iterations continue until Δx_k becomes small enough to signal convergence. TROLL uses a convergence test of the form:

$$\max_i \left(\frac{\Delta x_i^{(k)}}{\min(|x_i^{(k)}|, |x_i^{(k+1)}|) + \gamma} \right) < \varepsilon$$

where ε is a small convergence criterion, say 10^{-6}, and γ is a small positive number to avoid division by zero or instability when x_i is close to zero.

The TROLL simulator actually has three Newton algorithms, one for backward-looking models and two for forward-looking models. The backward-looking algorithm generates the Jacobian matrix by taking the partial derivatives with respect to the unlagged variables and decomposes it into the smallest truly simultaneous blocks, which can then be solved one-by-one. The "OLDSTACK" Stacked-Time algorithm for forward-looking models constructs a Jacobian for the system stacked over a specified time horizon, including derivatives with respect to the lags and leads; it then decomposes the stacked Jacobian into minimal simultaneous blocks and proceeds like the backward-looking algorithm. The "NEWSTACK" Stacked-Time algorithm uses a method developed by Laffargue [1990], Boucekkine [1995] and Juillard [1996] that takes advantage of the repetitive structure of the stacked system (see Hollinger [1996], Juillard et al. [1998]).

3. An example in one dimension

For a simple example, consider the single equation log(x) = 0 and a starting point $x^{(0)} = 2$:

Iteration	$x^{(k)}$	$F(x^{(k)}) = \log(x^{(k)})$	$J(x^{(k)}) = 1/x^{(k)}$	$\Delta x^{(k)} = -x\log(x)$
0	2.000000000000	0.693147180560	0.500000000000	-1.386294361120
1	0.613705638880	-0.488239881296	1.629445676635	0.299635568278
2	0.913341207158	-0.090645747330	1.094881072006	0.082790496290
3	0.996131703448	-0.003875797762	1.003883318380	0.003860805027
4	0.999992508475	-0.000007491553	1.000007491581	0.000007491497
5	0.999999999972	-0.000000000028	1.000000000028	0.000000000028
6	1.000000000000	0.000000000000	1.000000000000	0.000000000000

Here we see Newton's method converge to a very accurate solution in six iterations. Judging by the leading 9's in the $x^{(k)}$ column, quadratic convergence starts in the second iteration.

4. A numerical obstacle

If we start at $x^{(0)} = 3$, however, we have a very different result:

Iteration	$x^{(k)}$	$F(x^{(k)}) = \log(x^{(k)})$	$J(x^{(k)}) = 1/x^{(k)}$	$\Delta x^{(k)} = -x\log(x)$
0	3.000000000000	1.098612288668	0.333333333333	-3.295836866004
1	-0.295836866004	Illegal!		

The first Newton step takes x into a region where the residual $F(x)$ cannot be calculated, so Newton's method breaks down.

5. Damped Newton

One solution to this problem is to take a fraction of the Newton step each iteration:

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x^{(k)} = x^{(k)} - \alpha J(x^{(k)})^{-1} F(x^{(k)})$$

where α is a damping factor in the range (0,1]. For example, with $\alpha = 0.5$ and $x^{(0)} = 3$:

Iteration	$x^{(k)} = x^{(k-1)} + \alpha\Delta x^{(k-1)}$	$F(x^{(k)}) = \log(x^{(k)})$	$J(x^{(k)}) = 1/x^{(k)}$	$\Delta x^{(k)} = -x\log(x)$
0	3.000000000000	1.098612288668	0.333333333333	-3.295836866004
1	1.352081566998	0.301645306421	0.739600349867	-0.407849058583
2	1.148157037706	0.138158080969	0.870960998504	-0.158627172981
3	1.068843451216	0.066577177180	0.935590706817	-0.071160579830
4	1.033263161301	0.032721912099	0.967807657771	-0.033810346339
5	1.016357988132	0.016225637619	0.983905288961	-0.016491056406
6	1.008112459928	0.008079730815	0.991952822477	-0.008145277308
7	1.004039821275	0.004031683107	0.995976433216	-0.004047970386
8	1.002015836081	0.002013807010	0.997988219339	-0.002017866515
9	1.001006902824	0.001006396237	0.998994110010	-0.001007409580
10	1.000503198034	0.000503071472	0.999497055047	-0.000503324617
11	1.000251535725	0.000251504096	0.999748527529	-0.000251567358
12	1.000125752046	0.000125744140	0.999874263765	-0.000125759953
13	1.000062872070	0.000062870094	0.999937131883	-0.000062874046
14	1.000031435047	0.000031434553	0.999968565941	-0.000031435541
15	1.000015717276	0.000015717153	0.999984282971	-0.000015717400
16	1.000007858576	0.000007858546	0.999992141485	-0.000007858607
17	1.000003929273	0.000003929265	0.999996070743	-0.000003929280
18	1.000001964633	0.000001964631	0.999998035371	-0.000001964634
19	1.000000982315	0.000000982315	0.999999017686	-0.000000982316
20	1.000000491157	0.000000491157	0.999999508843	-0.000000491158
21	1.000000245579	0.000000245579	0.999999754421	-0.000000245579

Now the iterations converge legally toward the solution, but the fixed damping factor destroys the quadratic convergence of the full Newton step; many iterations are required.

Furthermore, a fixed damping factor that succeeds in some cases may easily fail in others. For example, keeping $\alpha = 0.5$ but with $x^{(0)} = 10$:

Iteration	$x^{(k)} = x^{(k-1)} + \alpha\Delta x^{(k-1)}$	$F(x^{(k)}) = \log(x^{(k)})$	$J(x^{(k)}) = 1/x^{(k)}$	$\Delta x^{(k)} = -x\log(x)$
0	10.000000000000	2.302585092994	0.100000000000	-23.025850929941
1	-1.512925464970	Illegal!		

6. Backtracking

A much more effective approach is to backtrack when the Newton step is illegal: apply damping only when needed and adjust the damping factor so the step is legal. The TROLL simulator uses a simple strategy of cutting the step in half until it finds a legal step.

With the simple log(x) example, $x^{(0)} = 2$ leads to the undamped Newton iterations:

```
SIMULATE Command: conopt concr 1e-9 stop 20;
SIMULATE Command: print eq all;

   1:          LOG(X) = 0

SIMULATE Command: list iterations x; list residuals all; list damp;
SIMULATE Command: simstart 2a; dosim 1;
Date:  Block: Iter:  What:            Value:           [Rel. Change:]
2A          1     0  X                    2
                     Eqn    1             0.6931471806 (Residual)
                  1  X                    0.6137056389 [-8.59075e-001]
                     Eqn    1            -0.4882398813 (Residual)
                  2  X                    0.9133412072 [+1.85682e-001]
                     Eqn    1            -0.0906457473 (Residual)
                  3  X                    0.9961317034 [+4.32701e-002]
                     Eqn    1            -0.0038757978 (Residual)
                  4  X                    0.9999925085 [+1.93414e-003]
                     Eqn    1        -7.49155322e-006 (Residual)
                  5  X                    1            [+3.74576e-006]
                     Eqn    1        -2.80615531e-011 (Residual)
                  6  X                    1            [+1.40308e-011]
                     Eqn    1             0            (Residual)
```

From $x^{(0)} = 3$, backtracking is required once, followed by undamped Newton iterations:

```
SIMULATE Command: simstart 3a; dosim 1;
Date:  Block: Iter:  What:            Value:           [Rel. Change:]
3A          1     0  X                    3
                     Eqn    1             1.0986122887 (Residual)

WARNING 15046
Expression cannot be evaluated.
In residual for equation 1:
LOG'F(X'N)
Date: 3A;  Block: 1;  Iteration: 1;
  -- attempt to take log of a non-positive number
(This WARNING will not be repeated during this block.)

                  1  X                    1.352081567  [-2.54340e+000]
                     Damping Factor:      0.5
                     Eqn    1             0.3016453064 (Residual)
                  2  X                    0.9442325084 [-2.09774e-001]
                     Eqn    1            -0.0573828419 (Residual)
                  3  X                    0.9984152531 [+2.78684e-002]
                     Eqn    1            -0.0015860039 (Residual)
                  4  X                    0.9999987436 [+7.92373e-004]
                     Eqn    1        -1.25637595e-006 (Residual)
                  5  X                    1            [+6.28188e-007]
                     Eqn    1        -7.89257548e-013 (Residual)
                  6  X                    1            [+3.94629e-013]
                     Eqn    1             0            (Residual)
```

When we start at $x^{(0)} = 10$, we have to backtrack the first two iterations as indicated by the lines listing "`Damping Factor`".   Convergence requires one extra iteration:

```
SIMULATE Command: delist res all;
SIMULATE Command: simstart 10a; dosim 1;
Date:  Block: Iter:  What:            Value:           [Rel. Change:]
10A         1     0  X                   10

WARNING 15046
Expression cannot be evaluated.
In residual for equation 1:
LOG'F(X'N)
Date: 10A;  Block: 1;  Iteration: 1;
  -- attempt to take log of a non-positive number
(This WARNING will not be repeated during this block.)

                  1  X                    4.2435372675 [-2.09326e+000]
                     Damping Factor:      0.25
                  2  X                    1.1767388621 [-2.12231e+000]
                     Damping Factor:      0.5
                  3  X                    0.9852282175 [-9.64678e-002]
                  4  X                    0.999890356  [+7.38562e-003]
                  5  X                    0.999999994  [+5.48220e-005]
                  6  X                    1            [+3.00556e-009]
                  7  X                    1            [+0.00000e+000]
```

Even $x^{(0)} = 100$ is handled effectively, with a damping factor as low as 0.125:

```
SIMULATE Command: simstart 100a; delist res all; dosim 1;
Date:  Block: Iter:  What:            Value:           [Rel. Change:]
100A        1     0  X                  100

WARNING 15046
Expression cannot be evaluated.
In residual for equation 1:
LOG'F(X'N)
Date: 100A;  Block: 1;  Iteration: 1;
  -- attempt to take log of a non-positive number
(This WARNING will not be repeated during this block.)

                  1  X                   42.4353726751 [-4.55957e+000]
                     Damping Factor:      0.125
                  2  X                    2.6736165135 [-3.66169e+000]
                     Damping Factor:      0.25
                  3  X                    0.0442963305 [-2.51779e+000]
                  4  X                    0.1823615004 [+1.32209e-001]
                  5  X                    0.4926977907 [+2.62472e-001]
                  6  X                    0.8414585008 [+2.33645e-001]
                  7  X                    0.9867098743 [+7.88784e-002]
                  8  X                    0.9999112924 [+6.64486e-003]
                  9  X                    0.9999999961 [+4.43538e-005]
                 10  X                    1            [+1.96732e-009]
                 11  X                    1            [+0.00000e+000]
```

This example is artificially trivial, but the problem it illustrates is common in real TROLL models.

7. A more realistic example: CAMK3

For a more realistic example, I used the Canadian submodel of the IMF's MULTIMOD Mark 3 (Laxton et al. [1998]). This is an annual forward-looking model with 92 equations, a maximum lag of 3 periods, and a maximum lead of 10 periods.

I applied a severe shock (permanent 120% increase in money target). That caused the first iteration to step out of bounds when I stacked it for 100 periods using NEWSTACK. After one backtrack, it took several iterations for the Newton steps to settle down:

```
TROLL Command: simulate stack 100;
Constructing stacked-time incidence matrix and code.
Simulations can start from 1977A to 2150A and must end by 2150A.
SIMULATE Command: list worst; list damp;   // Show 'worst' variable
SIMULATE Command: conopt stop 100 concr 1e-9 gamma 1e-5 divcr 1e6 ;
SIMULATE Command: simstart 2001a; dostack 1;
Date:  Block: Iter:  What:           Value:           [Rel. Change:]

WARNING 15046
Expression cannot be evaluated.
In residual for equation 17:
LOG'F(POIL'X/CA_ER'N/CA_PGNP'N)
Date: 2008A;  Block: 1;  Iteration: 1;
  -- attempt to take log of a non-positive number
(This WARNING will not be repeated during this block.)

2001A       1     1  CA_NEER[2010A]      0.0048071041 [-2.13539e+002]
                     Damping Factor:     0.5
                  2  CA_RL[2091A]        0.0022702907 [-1.67623e+003]
                  3  CA_RCI[2002A]    1.63126124e-005 [-1.96513e+003]
                  4  CA_TB[2015A]       -0.3014075726 [-1.21705e+002]
                  5  CA_TB[2018A]      -10.9987681355 [-8.33338e+001]
                  6  CA_TB[2017A]       -0.9700580732 [-4.56872e+000]
                  7  CA_TB[2016A]       -0.1527930314 [-6.64796e+000]
                  8  CA_TB[2016A]       -0.9104147398 [-4.95816e+000]
                  9  CA_TB[2029A]        0.0468686727 [-2.83773e+000]
                 10  CA_TB[2029A]        0.0370579734 [-2.64668e-001]
                 11  CA_TB[2029A]        0.0370111934 [-1.26360e-003]
                 12  CA_TB[2029A]        0.0370111924 [-2.84293e-008]
                 13  CA_TB[2029A]        0.0370111924 [-4.27052e-011]
```

A fixed damping factor also worked, but very slowly:

```
SIMULATE Command: conopt damp 0.5 ;      // Fixed damping
SIMULATE Command: simstart 2001a; dostack 1;
Date:  Block: Iter:  What:           Value:           [Rel. Change:]
2001A       1     1  CA_NEER[2010A]      0.519126063  [-2.13539e+002]
                  2  CA_RL[2091A]        1.913417055  [-1.67623e+003]
                  3  CA_RS[2012A]      -3.6681643773  [-1.34551e+002]
                  4  CA_RS[2014A]       2.1766276736  [-4.32089e+002]
                  5  CA_RCI[2031A]      0.0018116942  [-4.79506e+001]
                  6  CA_RCI[2014A]     -0.0114351245  [-2.23584e+001]
                  7  CA_RL[2094A]      -0.0587771369  [-2.07435e+001]
                  8  CA_RCI[2031A]      0.0001809989  [-8.89615e+000]
                  9  CA_RCI[2031A]    7.99601934e-005 [-6.50215e+000]
                 10  CA_RCI[2031A]    3.87710300e-005 [-6.63371e+000]
                 11  CA_RCI[2031A]    2.71003871e-005 [-9.17873e-001]
                 12  CA_DLGDP[2002A]   -0.0017650013  [-3.48021e-001]
                 13  CA_RCI[2031A]    3.44749060e-005 [+3.00934e-001]
                 14  CA_RCI[2031A]    4.01287657e-005 [+2.54249e-001]
                 15  CA_RCI[2031A]    4.41513589e-005 [+1.60490e-001]
                 16  CA_RCI[2031A]    4.65911851e-005 [+9.01114e-002]
                 17  CA_RCI[2031A]    4.79438404e-005 [+4.78045e-002]
                 18  CA_RCI[2031A]    4.86575578e-005 [+2.46348e-002]
                 19  CA_RCI[2031A]    4.90243726e-005 [+1.25070e-002]
                 20  CA_RCI[2031A]    4.92103532e-005 [+6.30183e-003]
                 21  CA_RCI[2031A]    4.93039960e-005 [+3.16305e-003]
                 22  CA_RCI[2031A]    4.93509825e-005 [+1.58460e-003]
                 23  CA_RCI[2031A]    4.93745173e-005 [+7.93073e-004]
                 24  CA_RCI[2031A]    4.93862952e-005 [+3.96731e-004]
                 25  CA_RCI[2031A]    4.93921866e-005 [+1.98411e-004]
                 26  CA_RCI[2031A]    4.93951330e-005 [+9.92181e-005]
                 27  CA_RCI[2031A]    4.93966064e-005 [+4.96118e-005]
                 28  CA_RCI[2031A]    4.93973431e-005 [+2.48068e-005]
                 29  CA_RCI[2031A]    4.93977114e-005 [+1.24034e-005]
                 30  CA_RCI[2031A]    4.93978956e-005 [+6.20183e-006]
                 31  CA_RCI[2031A]    4.93979877e-005 [+3.10087e-006]
                 32  CA_RCI[2031A]    4.93980338e-005 [+1.55044e-006]
                 33  CA_RCI[2031A]    4.93980568e-005 [+7.75222e-007]
                 34  CA_RCI[2031A]    4.93980683e-005 [+3.87611e-007]
                 35  CA_RCI[2031A]    4.93980741e-005 [+1.93807e-007]
                 36  CA_RCI[2031A]    4.93980769e-005 [+9.69066e-008]
                 37  CA_RCI[2031A]    4.93980784e-005 [+4.84512e-008]
                 38  CA_RCI[2031A]    4.93980791e-005 [+2.42324e-008]
                 39  CA_RCI[2031A]    4.93980795e-005 [+1.21113e-008]
                 40  CA_RCI[2031A]    4.93980796e-005 [+6.05415e-009]
                 41  CA_RCI[2031A]    4.93980797e-005 [+3.00819e-009]
                 42  CA_RCI[2031A]    4.93980798e-005 [+1.53058e-009]
                 43  CA_RCI[2031A]    4.93980798e-005 [+7.50337e-010]
```

8. Sometimes damping is necessary: JAXXX

An experimental version of the Japan submodel from MULTIMOD provides a tougher case. This model has 87 equations, a maximum lag of 3 periods, and a maximum lead of 50 periods. Attempting to solve 360 periods with NEWSTACK (total of 31320) equations led to explosive iterations despite backtracking:

```
TROLL Command: conopt concr 1e-3 stop 25 divcr 1e4 gamma 1e-5 ;
TROLL Command: simulate newstack 360;
Constructing stacked-time incidence matrix and code.
Simulations can start from 2001A to 2370A and must end by 2370A.
SIMULATE Command: list worst; list norm; list damp;
SIMULATE Command: simstart 2001a; dostack 1;
Date:  Block: Iter:  What:               Value:       [Rel. Change:]
2001A       1     0  Residuals Norm:  69191449.810819

WARNING 15046
Expression cannot be evaluated.
In residual for equation 21:
LOG'F(POIL'X/JA_ER'N/JA_PGNP'N)
Date: 2097A;  Block: 1;  Iteration: 1;
  -- attempt to take log of a non-positive number
(This WARNING will not be repeated during this block.)

                  1  JA_COIL[2334A]          0.009983 [-9.11362e+003]
                     Residuals Norm:  35188334.94299
                     Damping Factor:         0.5
                  2  JA_RSR[2349A]          -1.275466 [-3.72709e+003]
                     Residuals Norm:  30713316.081444
                     Damping Factor:         0.125
                  3  RES_JA_ICOM[2254A]     -0.000105 [+7.67232e+002]
                     Residuals Norm:   3369807.630321
                  4  RES_JA_ICOM[2254A]     -0.116919 [-1.01409e+003]
                     Residuals Norm:   2959782.57105
                     Damping Factor:         0.125
                  5  RES_JA_ICOM[2270A]     -0.116399 [-1.49599e+003]
                     Residuals Norm:   2666060.989729
                     Damping Factor:         0.5
                  6  RES_JA_ICOM[2321A]     -0.048376 [-8.06889e+002]
                     Residuals Norm:   1251557.492046
                  7  JA_W[2348A]        -20485.057986 [-3.57727e+002]
                     Residuals Norm:   3124021.613035
                  8  JA_WHTTNEW[2346A]       0.007488 [+2.88739e+002]
                     Residuals Norm:   3583314.066041
                     Damping Factor:         0.5
                  9  JA_PIMA[2348A]         -1.187478 [-9.05448e+002]
                     Residuals Norm:  10923240.761615
                     Damping Factor:         0.5
                 10  JA_VAT[2352A]           3.825236 [+2.20613e+004]
                     Residuals Norm:  1.34981162e+010
                     Damping Factor:         0.0625

ERROR 15011
Divergence occurred:
Date: 2352A;  Block: 1;  Iteration: 10
Worst Variable: JA_VAT
Last iteration value: 3.82524;  Relative Change: 22061.3
```

(Increasing the divergence criterion and raising the iteration limit to 100 did not help.)

In this case, setting a fixed damping factor of 0.8 allowed the iterations to settle down, and the model converged normally.

```
SIMULATE Command: conopt damp 0.8 ;
SIMULATE Command: simstart 2001a; dostack 1;
Date:  Block: Iter:  What:              Value:       [Rel. Change:]
2001A       1     0  Residuals Norm:  69191449.810819

WARNING 15046
Expression cannot be evaluated.
In residual for equation 21:
LOG'F(POIL'X/JA_ER'N/JA_PGNP'N)
Date: 2140A;  Block: 1;  Iteration: 1;
  -- attempt to take log of a non-positive number
(This WARNING will not be repeated during this block.)

                  1  JA_COIL[2334A]         18.223923 [-8.84302e+003]
                     Residuals Norm:  41544158.741429
                     Damping Factor:          0.4
                  2  JA_RCI[2008A]           0.098636 [+9.79447e+001]
                     Residuals Norm:   8206381.339372
                  3  JA_NFA[2008A]        -452.150283 [-4.76013e+001]
                     Residuals Norm:   1627375.614847
                  4  JA_TB[2028A]          -0.262267 [+1.37465e+001]
                     Residuals Norm:    325452.821132
                  5  JA_TB[2028A]           0.081109 [+2.57074e+000]
                     Residuals Norm:     65343.606619
                  6  JA_TB[2028A]           0.150401 [+1.06774e+000]
                     Residuals Norm:     14923.668466
                  7  JA_TB[2028A]           0.16429  [+1.15431e-001]
                     Residuals Norm:      1441.137986
                  8  JA_TB[2028A]           0.167068 [+2.11353e-002]
                     Residuals Norm:          6.527762
                  9  JA_TB[2028A]           0.167624 [+4.15678e-003]
                     Residuals Norm:          0.00014
                 10  JA_TB[2028A]           0.167763 [+8.28600e-004]
                     Residuals Norm:          0.00014
                 11  JA_TB[2028A]           0.167757 [+1.65618e-004]
                     Residuals Norm:         21.75886
                 12  JA_TB[2028A]           0.167762 [+3.31192e-005]
                     Residuals Norm:          0.0104
                 13  JA_TB[2028A]           0.167763 [+6.62366e-006]
                     Residuals Norm:  2.26581510e-007
                 14  JA_TB[2028A]           0.167763 [+1.32474e-006]
                     Residuals Norm:  5.33652831e-008
                 15  JA_TB[2028A]           0.167763 [+2.64937e-007]
                     Residuals Norm:  5.33652831e-008
```

9.   An extensive experiment: stochastic simulation of QPM

A robust and efficient solution algorithm is particularly important for applications like stochastic simulation, which must automatically solve many trials with random shocks.

To see whether backtracking could help in this context, I used a stochastic simulation macro developed at the Research Department of the Bank of Canada.   The macro works with a version of their Quarterly Projection Model, a quarterly with 451 equations, leads up to 20 periods, and lags up to 19 periods (Coletti et al. [1996]; Maclean and Pioro [2000]).

In order to simulate the effect of unanticipated shocks on a forward-looking model, each replication requires a separate stacked solution for each date, with the shock being introduced one period at a time. (This is a simplification of the actual process, since the shocks must also be used to generate a new steady-state solution.)

In this case, each replication covered 109 quarters and performed 109 stacked simulations using OLDSTACK; the stack horizon was limited to 20 periods to reduce the expense. Sixteen different experiments were performed using different parameter settings in the model, and 100 replications were run for each experiment. The same sets of random shocks were used in the 100 replications for each experiment. Solving the model was much easier in some experiments than others, and some sets of shocks were more difficult than others.

This macro was developed before the automatic backtracking was available, and it includes its own logic to retry when a simulation fails. If one of the stacked simulations fails to converge, the macro gets new starting guesses by running a backward-looking simulation (treating leads as exogenous), and it also applies a fixed damping factor for the next stacked simulation. The retries increase the chance of success significantly, but at some expense in simulation time. As the following table shows, automatic backtracking eliminated the need for retries in a significant number of replications, 176 out of 1600. Another 126 replications required a retry to find a solution; I do not yet know if any of those required the fixed damping factor — perhaps the improved starting guess from the backward-looking simulation would suffice.

Experiment	No backtrack or retry needed	OK with backtrack, no retry	Needed retry even with backtrack	Failed even with retry	Total
1	46	19	35	0	100
2	58	28	14	0	100
3	63	19	12	6	100
4	77	16	7	0	100
5	58	16	13	13	100
6	18	6	0	76	100
7	0	0	0	100	100
8	78	13	9	0	100
9	77	12	11	0	100
10	73	14	11	2	100
11	62	17	6	15	100
12	37	12	7	44	100
13	14	4	1	81	100
14	0	0	0	100	100
15	0	0	0	100	100
16	0	0	0	100	100
Total	**661**	**176**	**126**	**637**	**1600**

10. Conclusion and future work

Automatic backtracking is a significant aid in solving nonlinear models robustly. However, there are cases where backtracking alone is insufficient to bring the iterations into the region of convergence. Sometimes there are ways to improve the starting guesses, which can help in such cases; the use of backward-looking simulation in preparation for forward-looking simulation is one example.

Perhaps more useful will be to incorporate a search along the Newton step even when the full step is legal. An algorithm recommended by Dennis and Schnabel [1996] uses the sum of squared residuals (SSR) to judge the quality of the Newton step and to backtrack when the full step goes too far. My experiments with this algorithm have so far proved disappointing with real TROLL models. One problem is that the objective function used — the SSR — is highly dependent on the scaling of the residuals, which can vary over many orders of magnitude. Nowak and Weimann [1991] describe an algorithm that is somewhat more complicated but provides automatic scaling of the residuals; that algorithm may prove more practical in actual use.

11. References

Boucekkine, R. (1995), “An alternative methodology for solving nonlinear forward-looking models”, *Journal of Economic Dynamics and Control*, 19:711-734.

Coletti, Donald; Benjamin Hunt, David Rose, and Robert Tetlow, (1996), “The Bank of Canada’s New Quarterly Projection Model; Part 3, The Dynamic Model”, *QPM Technical Report no. 75*, Bank of Canada, Ottawa.

Dennis, J. E., Jr.; and Robert B. Schnabel, (1996), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia.

Hollinger, Peter, (1996), “The Stacked-Time Simulator in TROLL: A Robust Algorithm for Solving Forward-Looking Models”, presented at the *Second International Conference on Computing in Economics and Finance*, Geneva, Switzerland, 26–28 June 1996, Intex Solutions, Inc., Needham, MA.

Juillard, Michel (1996) “DYNARE: A program for the resolution and simulation of dynamic models with forward variables through the use of a relaxation algorithm”, CEPREMAP 9602, Paris.

Juillard, Michel; Douglas Laxton, Peter McAdam, and Hope Pioro, (1998), “An Algorithm Competition: First-Order Iterations Versus Newton-Based Techniques”, *Journal of Economic Dynamics and Control* 22:1291-1318.

Laffargue, J.-P. (1990), “Résolution d’un modèle macroéconometrique avec anticipation rationelles”, *Annales d’Economie et Statistique*, 17:97-119.

Laxton, Douglas; Peter Isard, Hamid Faruqee, Eswar Prasad, and Bart Turtelboom, (1998), “MULTIMOD Mark III: The Core Dynamic and Steady-State Models,” Occasional Paper 164, International Monetary Fund, Washington, DC.

Maclean, Dinah; and Hope Pioro, (2000), “Price Level Targeting – The Role of Credibility”, Research Department, Bank of Canada.

Nowak, U.; and L. Weimann, (1991), “A Family of Newton Codes for Systems of Highly Nonlinear Equations”, TR-91-10, Konrad-Zuse-Zentrum fűr Informationstechnik, Berlin