

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/228467394>

# A Recursive Algorithm to Generate Piecewise Linear Budget Constraints

Article · October 2003

CITATIONS

0

READS

73

2 authors:



**Alan S Duncan**

Curtin University

132 PUBLICATIONS 2,753 CITATIONS

[SEE PROFILE](#)



**Graham Stark**

Virtual Worlds Research

20 PUBLICATIONS 111 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Empirical immigration research [View project](#)



Non-parametric methods [View project](#)

# A Recursive Algorithm to Generate Piecewise Linear Budget Constraints\*

Alan Duncan<sup>†</sup> and Graham Stark<sup>‡</sup>

September 2003

## Abstract

This paper introduces a recursive algorithm with which to generate complete and accurate budget constraints using static microsimulation models. We describe the generic algorithm, and discuss an extension by which reasons for any change in the marginal effective tax rate may be explained. We show how the recursive algorithm allows for the derivation of continuous expressions for average tax rate (ATR), average marginal tax rate (AMTR), replacement rate (RR) and marginal effective tax rate (METR) schedules, expressed solely in terms of the marginal wage rate and ‘virtual’ income. A practical application serves to highlight the usefulness of this algorithm in fiscal policy analysis at the micro level.

**JEL Classifications: C63; D12**

---

\*We are grateful to John Creedy, Chris Giles, Julian McCrae and Jocelyn Payne for useful comments and suggestions. The usual disclaimer applies.

<sup>†</sup>School of Economics, University of Nottingham, University Park, Nottingham NG7 2RD, United Kingdom and Institute for Fiscal Studies. Tel: +44 115 846 6107; Fax: +44 115 951 4159; e-mail: alan.duncan@nottingham.ac.uk

<sup>‡</sup>Institute for Fiscal Studies, 7 Ridgmount Street, London WC1E 7AE, United Kingdom. Tel: +44 171 291 4800; e-mail: g\_stark@ifs.org.uk

# 1 Introduction

Tax microsimulation modelling is now commonly used in public policy analysis to assess the distributional impact of a tax or benefit change, and to simulate the likely cost to the Government.<sup>1</sup> Indeed, the microsimulation method is perhaps the only way to generate realistic estimates of the effects of tax or benefit reform on a heterogeneous population. The typical approach adopted by tax microsimulation practitioners is to model the impact of tax benefit changes on each tax unit in a large and representative sample of households, and then to aggregate individual effects over the sample to simulate the effect over the full population.<sup>2</sup> In some cases, these simulations are conditioned on observed patterns of (principally labour supply) behaviour, and hence have not been adapted to give a full description of the budget constraint for a particular tax unit over some feasible range of income. More recently, tax microsimulation models have been adapted to model employment responses following some reform to the tax and transfer system. These *second-generation* microsimulation models generally work by placing individuals at some optimal point on their modelled budget constraint under alternative tax and transfer scenarios. However, even among these second-generation studies one tends to find that the full budget constraint which faces an individual or family is approximated by repeatedly evaluating net incomes at incremental gross income levels. This is an inefficient method which requires many unnecessary evaluations of net income, and can miss the exact point on the budget constraint at which marginal effective tax rates (METRs) change.

The absence of a full and detailed budget constraint generator limits the power of tax microsimulation methods in a number of respects; as a commentary on tax and transfer interactions; as a means to generate summary measures of incentives; as an input into the behavioural microsimulation of the effects of tax reform on labour supply. The complexity of many countries' tax and transfer systems can lead to difficulties in the interpretation of the relationship between gross and net household income.<sup>3</sup> Taxes and means-tested welfare payments can combine to produce marginal

---

<sup>1</sup>For examples of the tax microsimulation approach, see *inter alia* ? for a UK application; ? for the US; ? for Canada; ? for Ireland; ? for Australia.

<sup>2</sup>Some countries also apply behavioural microsimulation methods to model employment incentive effects, although this is less common.

<sup>3</sup>The UK system, for example, includes two direct taxes (Income Tax and National Insurance), four means-tested benefits (Working Families' Tax Credit (WFTC), Housing Benefit, Council Tax Benefit and Income Support), and a number contingent benefits (such as Child Benefit, Invalidity

tax rates close to 100% (or even - though this is rarer - in excess of 100%, so families *lose* income as they work more). There can be startling discontinuities as particular taxes or welfare payments kick in or are stopped as earnings or hours worked exceed some level. With systems as complex as this, the availability of a generic algorithm to generate exact and authentic budget constraints is an extremely valuable tool.<sup>4</sup>

In this short paper we describe a generic method by which microsimulation models may generate complete and accurate budget constraints for each member of the simulation sample, allowing for the fact that each individual's constraint is unique. Our method takes advantage of recursions which exploit the geometric piecewise linear nature of the typical budget constraint. The result is an algorithm which is efficient, and which identifies exactly the points on a budget constraint at which marginal effective tax rates change. The generic algorithm we develop may be adapted to produce fully continuous expressions for marginal effective tax rate (METR), average tax rate (ATR) and replacement rate (RR) measures at the level of the individual. These are valuable summary measures which give insight into the distortionary impact of the tax system, and the likely incentive impact of tax reform. We also discuss a further extension to the basic algorithm which serves automatically to identify the *reasons* for each change in marginal effective tax rates.

In Section 2 we set out the generic algorithm, including reference to sketched computer code to assist in the implementation of our algorithm elsewhere. We develop the output from the basic algorithm to generate continuous expressions for commonly used budget constraint measures (specifically, average and marginal tax rates and replacement rates). Section 4 reports a practical application. This serves to demonstrate the uses to which our budget constraint algorithm may be put when analysing tax policy at the micro level. Finally, Section 5 briefly concludes.

## 2 A recursive algorithm

We presume the existence of a static microsimulation procedure which can generate tax unit net incomes under different economic and socio-demographic circumstances.

---

Benefit etc.), a local property tax (the Council Tax), numerous means-tested or passported in-kind benefits (such as Legal Aid and free Dental Treatment), plus a complex system of indirect taxes.

<sup>4</sup>Indeed, some of the tax systems we looked at were so complicated that when we originally developed our algorithm most of our time was spent convincing ourselves that some highly unusual budget constraints generated by the algorithm could possibly be right.

In particular, our algorithm requires a tax microsimulation procedure of the form

$$y = f(h_k | H_{-k}, W, X, T) \quad (1)$$

where  $y$  represents tax unit net income as a function of the hours of work  $h_k$  (or equivalently gross income  $g_k$ ) for the  $k$ th member of the tax unit. Net income is conditioned on gross wages  $W = \{w_1, \dots, w_K\}$  for each person in the tax unit, hours of work  $H_{-k}$  for all people other than the  $k$ th member of the tax unit, individual and tax unit level characteristics  $X$ . Finally,  $T$  denotes the tax system on which tax unit net income is based.

One approach to calculating budget constraints, which is often used for illustrative purposes, involves using a static model to evaluate net income for a large number of labour supply (time) intervals. In order to identify the corners and discontinuities reasonably accurately, small time intervals are needed; for five-minute intervals spread over 50 hours of work, 600 evaluations of net income would be required. However, in the context of behavioural models involving a large number of different individuals, this would be computationally very cumbersome and inefficient. In particular, it requires the same number of evaluations of net income irrespective of the complexity of an individual's budget constraint, and may not precisely identify the position of each kink. Budget constraints can be obtained more efficiently and accurately using a recursive algorithm which exploits the piecewise linearity inherent in the majority of tax systems. The algorithm builds on two observations. First, each linear segment on a budget constraint can be described completely by extrapolating a line drawn between any two points on that segment. Second, the intersection point between two adjacent linear segments identifies exactly the point at which the marginal effective tax rate changes.<sup>5</sup>

Before moving to a description of the algorithm itself, it is important to understand a number of assumptions implicit in our implementation:

1. gross earnings per hour are constant. Thus, hours of work are just gross earnings divided by some hourly wage. This rules out overtime, on the one hand, and jobs where earnings are not directly related to hours of work (such as the author's jobs) on the other;

---

<sup>5</sup>Discontinuities in the budget constraint obviously complicate the intuition, but may also be accommodated within the same generic algorithm.

2. take-up of benefits is complete, and taxes are universally paid. There is some evidence<sup>6</sup> to suggest that small amounts of benefits are less likely to be claimed. So a real-world budget constraint might have yet more discrete jumps in it as small amounts of benefit were judged no longer worth claiming;
3. changes in gross earnings do not impact on other income or expenditure measures. Hence, work expenses, pension contributions etc. are all held constant, as are the earnings of other family members;
4. we also ignore all issues of the timing of transfer payments. There are a number of transfer programmes around the world for which entitlement remains fixed for a period of time following initial assessment.<sup>7</sup> Thus, the relationship between gross income (or hours of work) and the level of transfer payments in the short-run may well be different to that in the longer term. Our budget constraints should be interpreted as long-run representations of the relationship between gross and net income.

Each of these assumptions could be relaxed without breaking the algorithm, although the interpretation of results would inevitably be more difficult.

## 2.1 Convex budget sets

Consider the simple budget constraint shown in Figure 1 consisting of three linear sections. This represents a convex budget set associated with an increasing, or progressive, marginal rate structure. The algorithm begins by taking the minimum and maximum number of hours,  $h_0$  and  $h_{\max}$  respectively, adding a small increment,  $\Delta h$ , to the first and subtracting it from the second, and evaluating the net incomes corresponding to the resulting four values of  $h$ . These values can be used to construct the straight lines shown as AC and CE, which are found to intersect at point C. The net income corresponding to the labour supply at C is then evaluated and compared with the actual net income at C. The comparison reveals that C is not on the budget constraint.

The algorithm continues by considering the two separate ranges, between  $h_0$  and  $h_C$  and that between  $h_C$  and  $h_{\max}$ . This involves finding the equations of the budget

---

<sup>6</sup>see ?

<sup>7</sup>For example, National Insurance (NI) in the UK is recalculated weekly whilst the new Working Families' Tax Credit (WFTC) is paid for six months regardless of any change in circumstances.

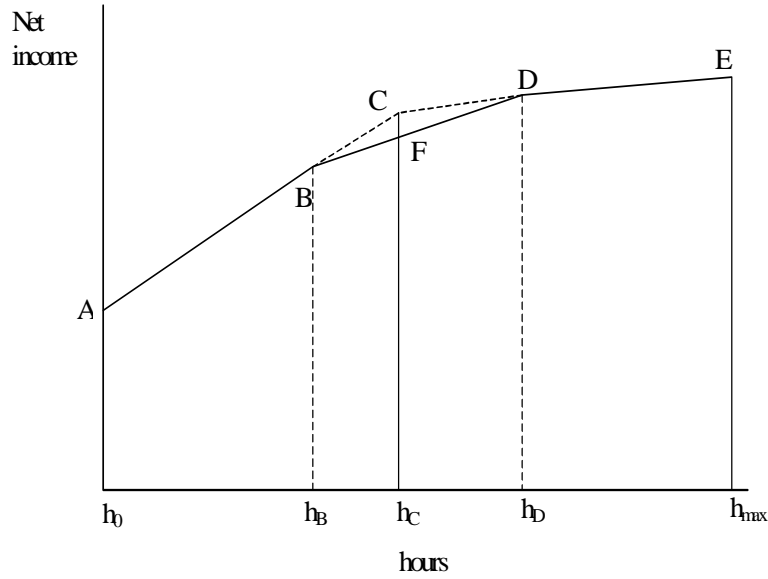


Figure 1:

lines either side of point F and the intersection with lines from A and E (the slopes at  $h_0$  and  $h_{\max}$  have of course already been computed). The intersection points, B and D, are then found to be on the budget constraint: the net incomes at those points are compared with the net incomes calculated for the hours corresponding to those points. Also, the slopes of these lines are the same. Hence BF and FD are found to be on the same segment of the budget constraint, so they form the line BD.

However, it is not appropriate to stop, because it cannot be assumed, without further checking, that the constraint between points A and B, and between D and E, are in fact straight lines; there may be other kinks and/or discontinuities (associated, for example, with the types of non-convex budget sets discussed in the next subsection). It is necessary to consider the lines obtained by moving to the left and right hand sides of each of the points B and D respectively. Comparisons would therefore reveal that the line on the left side of  $h_B$  coincides with the line from  $h_0$ , and the ‘intersection’ of the two relevant lines, a common line, defines the range AB of the true budget constraint. Similarly, the line on the right of  $h_D$  is found to coincide with that on the left of  $h_{\max}$ , which generates the linear section DE of the budget constraint. The algorithm therefore rapidly identifies the budget constraint as the piecewise linear constraint ABDE.<sup>8</sup>

<sup>8</sup>The procedure assumes that, where lines from two points are found to coincide, there are no

In this example, just 13 evaluations of net income are required to find the exact budget constraint. In the case of a budget constraint consisting of two linear sections, and hence only one kink, it can be seen that only seven evaluations of net income corresponding to seven different values of hours worked are required. This algorithm is obviously much more efficient than simply ‘moving across’ the complete range of hours by taking small increments and evaluating net incomes for each value.

## 2.2 Non-convex budget sets

Further complications arise in the case of non-convex budget sets. For example, consider Figure 2, showing the budget constraint ABCD. In this case, starting from  $h_0$  and  $h_{\max}$  produces an intersection point at E, which is associated with negative hours. The next stage is to divide the hours range,  $h_0$  to  $h_{\max}$ , into half, giving  $h_1$ . Consideration of the two separate sections to the right and left of F then reveals the intersection points B and C. Again, examination of the lines from B and C moving to the left and right of each, establishes the ranges AB, BF, FC and CD. Furthermore, the slopes of the sections BF and FC are equal, so they are indeed part of the same line. Hence discovery of the constraint of Figure 2 involves the evaluation of net income for 14 different hours levels. The important principle involved in this case is that when an intersection point occurs outside the relevant range of hours, it is necessary to divide the range into half.

Figure 3 shows an example of a non-convex budget constraint involving a discontinuity between B and C. Starting from the limits,  $h_0$  and  $h_{\max}$ , generates the intersection point E, corresponding to  $h_E$ , which is found to be on the true budget constraint. Examination of the line to the left of point E immediately generates the section AE as part of the constraint; the line from A coincides with that from E. But taking the right hand side of E in combination with the left hand side of D again gives E as the point of intersection. In cases where such a ‘fixed point’ occurs, it is again necessary to divide the relevant range, this time between  $h_E$  and  $h_{\max}$ , into half, giving hours of  $h_F$  and associated net income of F. Moving in a rightward direction from F identifies the length FD as being on the budget constraint, since the line from F coincides with that from D. However, moving to the left of F, combined with a move to the right of E, again produces the fixed point at E.

---

intermediate kinks involved. This is because it is improbable that any practical budget constraint would take such a form.



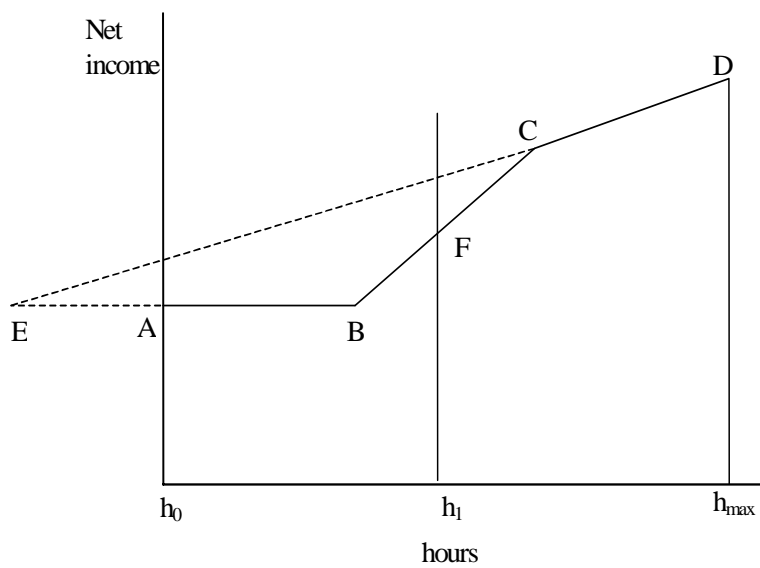


Figure 2:

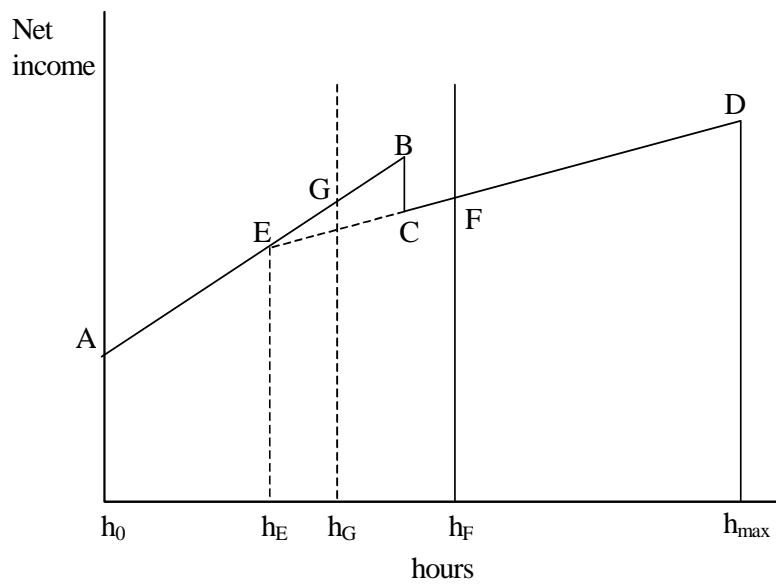


Figure 3:

The next step is therefore to divide the range between  $h_E$  and  $h_F$  in half, giving hours of  $h_G$ . Moving to the left of G gives the additional range EG, while moving to the right of G, and the left of F continues to generate the fixed point of intersection, E. Up to this stage, 13 evaluations of net income are required (remembering that there is no need to repeat the calculations once net income for a particular value of hours has been found). The procedure continues by dividing the range between  $h_G$  and  $h_F$ , into half and continuing as before. The algorithm iterates towards convergence at the discontinuity, where the number of iterations required depends on the level of accuracy specified.

The general rules therefore apply of dividing the relevant range of hours into two sub-ranges whenever a fixed point is observed or whenever an intersection outside the relevant range is observed. Whenever an intersection point is found in the feasible range, the lines either side of that point are investigated. Sections of the true budget constraint are identified when lines from two separate points are found to coincide.

In applying the algorithm, the position of all kinks and slopes of all linear sections of the budget constraint are automatically computed. In addition, the algorithm automatically generates the intersection with the vertical axis at zero hours for each linear segment. For each linear section, the slope is identified as the net wage rate and the intercept is referred to as the ‘virtual income’ corresponding to that section. The concept of virtual income plays a crucial role when examining labour supply behaviour. This is because an individual who is on a particular linear segment can be regarded as if optimisation is achieved with respect to a simple linear constraint having the appropriate net wage and with virtual income instead of the actual non-wage income.

### 2.3 JAVA implementation of the recursive algorithm

The algorithm has been implemented in several computer languages (Modula-2 for the initial development at the Institute for Fiscal Studies (IFS) in the United Kingdom<sup>9</sup>, subsequently GAUSS when developing a model of the Australian tax and transfer system<sup>10</sup>, and most recently JAVA for the Virtual Economy (VE) project, again at the IFS. Here, we show a slightly edited version of the main routine from the JAVA

---

<sup>9</sup>?

<sup>10</sup>The Melbourne Institute Tax and Transfer Simulator (MITTS, see ?).

version.<sup>11</sup>

All our implementations so far have used *recursion* - functions that call themselves. As is well known<sup>12</sup>, recursion allows the production of code that is easy to read and maintain, but often at the expense of efficiency: if efficiency was the prime concern, or if the chosen language does not support recursion (e.g. FORTRAN pre-F9x), our code could be re-implemented using GOTO statements at some cost to readability.<sup>13</sup>

The generator is structured so as to be independent of any particular model. As discussed in Section 2 above, the algorithm needs to see the target model as a simple function of the form

$$NetIncome = F(GrossIncome),$$

or

$$NetIncome = F(Hours).$$

Any simulation model that can be expressed in this way can be used. In an object-orientated language such as JAVA, you can hide much of the detail of a complex model inside an object; so long as the object has an accessible method with this signature, it can be used by the generator. Modula-2 is not object-orientated, but you can achieve similar independence by using procedure variables, though the code is messier and requires more extensive use of global variables. In JAVA, the entire budget constraint generator algorithm is then itself encapsulated in a class which takes the details of a particular household as input and returns the collection of  $(NetIncome, GrossIncome)$  points which make up the budget constraint.

### 2.3.1 A JAVA routine for recursive budget constraint generation

Our implementation presumes the existence of a function  $F(.)$  with a single argument (either hours or gross income). This represents the call to a standard (static) tax microsimulation model. Some specific control parameters are required. These include:

- **incr**: the increment to add or subtract from each income/hours point;

---

<sup>11</sup>Readers can experiment with the algorithm at the VE website at <http://ve.ifs.org.uk> - go to the Model section select the Advanced input form and scroll to the bottom to see the Budget Constraint options.

<sup>12</sup>See ? Chapter 5, pp51.

<sup>13</sup>See ? pp61 for an example.

- **start**, **stop**: the x-coordinate of the extreme points of the current segment

We also need some record or class representing a line and a point, and some data structure to hold the resulting collection of output points - in this example, we use a SortedMap from the *JAVA Collections Framework* to keep the points in order and to avoid duplication.

A sketch of the the main algorithm routine follows. For clarity, this is somewhat simplified compared to the JAVA version we have implemented on the Institute for Fiscal Studies' Virtual Economy website <http://ve.ifs.org.uk>, which includes more diagnostic and error handling code, as well as code to avoid making redundant recalculations:

```

1 void generateLineSegments(start,stop,incr) throws KinksException{
2
3 // halting condition 1
4 if( (start-stop) < incr ) {
5 return;
6 }
7
8 // create points, two at either end of a segment
9 Point[] p = new Point[ 4 ];
10 p[ 0 ] = F( start );
11 p[ 1 ] = F( start + incr );
12 p[ 2 ] = F( stop );
13 p[ 3 ] = F( stop - incr );
14
15 // generate the lines from the points
16 Line[ ] line = new Line[ 2 ];
17 line[ 0 ] = new Line( p[ 0 ] , p[ 1 ] );
18 line[ 1 ] = new Line( p[ 2 ] , p[ 3 ] );
19
20 // test the lines..
21 if ( line[ 0 ].sameLineAs( line[ 1 ] )) {
22
23 // halting condition 2: store points if lines are the same
24 points.put( p[ 0 ].getX( ) , p[ 0 ] );

```

```

25 points.put( p[ 2 ].getX( ) , p[ 2 ] );
26 return;
27
28 } else {
29
30 // carry on recursively
31 double splitPointX = 0;
32 Point ptmp = line[ 0 ].intersection( line[ 1 ] );
33 if ( (ptmp.x <= start ) || ( ptmp.x >= stop )) {
34 // if intersection below start or above stop, use midpoint
35 splitPointX = start - (stop- start)/2.0;
36 } else {
37 splitPointX = ptmp.x;
38 }
39 // commence recursive search on lower half of the current segment..
40 generateLineSegments( start , splitPointX );
41
42 // continue recursive search on upper half of the current segment..
43 generateLineSegments( splitPointX , stop );
44 }
45 }

```

Here  $F(\cdot)$  is our call to the tax microsimulation procedure. This returns a point  $(NetIncome, GrossIncome)$  where  $NetIncome$  is the tax unit net income at gross tax unit income  $GrossIncome$ . The routine may equally well be defined in terms of hours of work  $Hours$  rather than  $GrossIncome$ , since  $GrossIncome = W.Hours$  for (fixed) gross wage  $W$ . At lines 10-13 this procedure is called four times to create a series of points in  $(NetIncome, GrossIncome)$  space, and at lines 16-18 these points are used to generate the lines corresponding to AC and CE in Figure 1.

Lines 4-5 are a simple overflow check for cases where the budget constraint is near-vertical. Line 21 tests for whether the two line segments are in fact the same line; if so, as discussed in section 2.1, we add the points at either end to our budget constraint and return (lines 24-26).

If neither of these conditions are met, the algorithm continues (lines 30-43). Line 32 calculates point C in Figure 1. At lines 40-43 `GenerateLineSegments` is called

recursively to calculate the sub-budget constraints either side of C and so on until the halting conditions are met.

The initial call to `GenerateLineSegments` would have `start` and `stop` set to gross income at, say, 0 and 100 hours per week. With each recursive call, these ranges are shortened until a linear segment is generated.

### 3 Calculating average tax rates and replacement rates

Once the generic budget constraint generation routine is run, we are able to summarise the budget constraint completely in terms of the combinations of gross and net incomes at which marginal tax rates change (as in Tables 1 and 2). In fact, since the algorithm is essentially geometric in nature, the 'virtual income'  $\mu_k$  and marginal wage  $\omega_k$  are generated as intrinsic elements of the calculation. One significant advantage of our exact recursive algorithm is that, once we have summarised the  $k$  segment budget constraint in terms of the hours levels  $h_k$  at which virtual incomes  $\mu_k$  and marginal wage rates  $\omega_k$  change, we may generate continuous functions for the average tax rate (ATR), replacement rate (RR) and marginal effective tax rate (METR) with no requirement to access the tax and benefit model any further. Specifically, each measure can be expressed solely in terms of  $h_k$ ,  $\mu_k$  and  $\omega_k$ . As with the budget constraint generation routine itself, the solution to the derivation of continuous ATR and RR schedules is essentially geometric, and is set out in Figure 4;

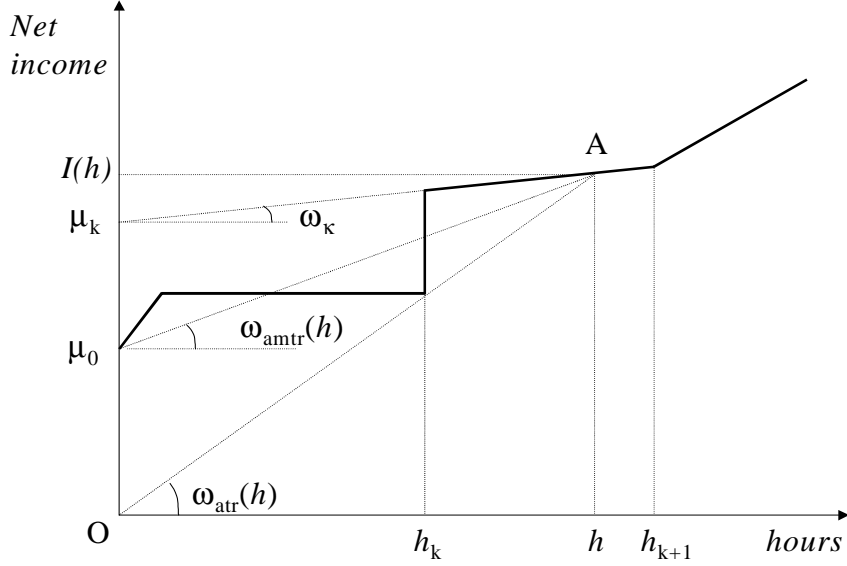
#### 3.1 the average tax rate (ATR) and average marginal tax rate (AMTR) schedules

We can define two average tax measures, the average tax rate  $ATR(h)$  and the average marginal tax rate  $AMTR(h)$ .<sup>14</sup> (some explanation here of the difference). The graphical representation of these two concepts is described in Figure 4. The average marginal wage  $\omega_{amtr}(h)$  is given by the line  $\mu_0A$ , and intersects the  $k$ th linearised budget constraint segment  $\mu_kA$  at net income  $I(h)$ , whereas the average wage  $\omega_{atr}(h)$  is given by the slope  $OA$ .

---

<sup>14</sup>We thank John Creedy for pointing out this distinction.

Figure 4: Deriving Average Tax Rates and Replacement Rates



Consider an arbitrary hours level  $h \in [h_k, h_{k+1})$  for some linear segment  $k$  of the budget constraint. Turning first to the average marginal tax rate, we have that

$$\begin{aligned}\mu_k + \omega_k \cdot h &= \mu_0 + \omega_{amtr}(h) \cdot h \\ &= \mu_0 + W \cdot [1 - AMTR_k(h)] \cdot h\end{aligned}$$

where  $AMTR_k(h)$  represents the average marginal tax rate at any point on the  $k$ th linearised budget segment, and  $W$  denotes the gross wage rate. This may be rearranged to give

$$\begin{aligned}AMTR_k(h) &= 1 - \frac{(\mu_k - \mu_0) + \omega_k \cdot h}{Wh} \\ &= \frac{(W - \omega_k) \cdot h - (\mu_k - \mu_0)}{Wh}\end{aligned}$$

where  $\mathbf{1}(\cdot)$  denotes the indicator function. So, the full average tax rate schedule  $ATR(h)$  is given by

$$AMTR(h) = \sum_k \mathbf{1}(h_k \leq h < h_{k+1}) \cdot AMTR_k(h) \quad (2)$$

The average tax rate  $ATR(h)$  is derived in a similar way to  $AMTR(h)$ . Specifically, the average tax rate satisfies

$$\begin{aligned} \mu_k + \omega_k \cdot h &= \omega_{atr}(h) \cdot h \\ &= W \cdot [1 - ATR_k(h)] \cdot h, \end{aligned}$$

where  $ATR_k(h)$  represents the average tax rate on the  $k$ th budget segment. This gives

$$\begin{aligned} ATR_k(h) &= 1 - \frac{\mu_k + \omega_k \cdot h}{Wh} \\ &= \frac{(W - \omega_k) \cdot h - \mu_k}{Wh}, \end{aligned}$$

leading to a full average tax rate schedule  $ATR(h)$  of the form

$$ATR(h) = \sum_k \mathbf{1}(h_k \leq h < h_{k+1}) \cdot ATR_k(h) \quad (3)$$

### 3.2 the replacement rate (RR) schedule

To calculate the replacement rate  $RR(h)$  for some arbitrary hours level  $h$ , we simply divide the intercept income  $\mu_0$  by total net income  $I(h)$  at any point on the budget constraint (see Figure 4). However, since  $I(h) = \mu_k + \omega_k \cdot h$  at hours  $h \in [h_k, h_{k+1})$ , the replacement rate  $RR_k(h)$  for the  $k$ th segment of the budget constraint is

$$RR_k(h) = \frac{\mu_0}{\mu_k + \omega_k \cdot h},$$

and the full replacement rate schedule  $RR(h)$  is

$$RR(h) = \sum_k \mathbf{1}(h_k \leq h < h_{k+1}) \cdot RR_k(h) \quad (4)$$



### 3.3 the marginal effective tax rate (METR) schedule

One can generate the marginal effective tax rate  $METR(h)$  straightforwardly in terms of  $\omega_k$  and  $h_k$  as follows:

$$METR(h) = \sum_k \mathbf{1}(h_k \leq h < h_{k+1}) \cdot \left[ \frac{W - w_k}{W} \right].$$

### 3.4 why do marginal effective tax rates change?

For many countries, the complexity of budget constraints facing particularly low-income households can be difficult to anticipate, and certainly difficult to explain. Most tax and transfer systems around the world have some or more of the following design features; a so-called 'stacking' of rates across transfer programs; the use of differential sources of assessable income for taxation and welfare benefit purposes; overlapping tapers; and the 'passporting' benefits (whereby the entitlement to one welfare programme creates an entitlement to another benefit). This makes the assessment of net household income corresponding to a specific value of individual gross income or hours or work complex in the extreme.

The ability to explain and rationalise marginal rate changes is an important descriptive and diagnostic tool in the face of these sorts of complexities. The recursive budget constraint algorithm we have developed lends itself to this sort of diagnostic application, since the points at which marginal rates change are pinpointed exactly. Once the constraint is generated, a second routine scans across the set of kink points looking for explanations for each one; this routine is one consequence of our initial skepticism of some of the shapes we were seeing. The particular algorithm for generating reasons for marginal effective tax rate changes requires that we collect information on entitlement to all benefits and payments of all sources of taxation across the kink point. Let  $h_k$  represent the  $k$ th kink point on a budget constraint, and suppose that net income  $F(\cdot)$  is evaluated at three successive points  $h_k - \Delta h$ ,  $h_k$  and  $h_k + \Delta h$  for some small  $\Delta h$ . In general terms, the following set of rules serve to explain the reason for a METR change:

1. if entitlement to a specific benefit moves from zero at  $h_k$  to some positive number at  $h_k + \Delta h$ , the kink has (at least in part) been caused by someone in the household moving onto a welfare programme at point  $h_k$ ;

2. if entitlement to a specific benefit moves from some positive number at  $h_k - \Delta h$  to zero at  $h_k$ , the kink can be explained in part by someone losing entitlement to a benefit;
3. if the increment in entitlement or tax payment between  $h_k - \Delta h$  and  $h_k$  is different to the increment between  $h_k$  and  $h_k + \Delta h$ , then the kink can be partially explained by a change in the withdrawal taper or tax rate.

Of course, any one turning point in a budget constraint may be caused by more than one of the rules listed above. The present implementation requires a certain amount of local knowledge of the tax system, and so can't easily be generalised. It may also fail to identify explanations in all cases. Nevertheless, the algorithm can greatly add to the diagnostic power of the tax microsimulation procedure at the level of the household, and has proved an invaluable debugging tool in programme development.

## 4 A practical application

We concentrate here on a practical implementation of the generic algorithm by presenting budget constraints, marginal effective tax rate (METR) schedules and METR diagnostics derived from a standard tax microsimulation model. This illustrative implementation is based on input from the Melbourne Institute Tax and Transfer Simulator (MITTS)<sup>15</sup>, although as has been pointed out, the budget constraint algorithm is designed to be independent of the tax microsimulation model itself.

We focus on the budget constraint facing the primary earner in a two-person household under the March 1999 Australian tax and transfer system. Figures 5 and 6 present graphical representations of the budget constraint and METR schedules facing the primary earner in a household with three children (two of whom are pre-school age) with a gross hourly wage of Aus\$25. The complexity of the tax schedule, and the success and accuracy with which the algorithm pinpoints kinks and discontinuities, is evident particularly from the second METR schedule.

---

<sup>15</sup>MITTS is a new microsimulation model of the Australian tax and transfer system under development at the Melbourne Institute of Applied Economic and Social Research. For full details, see ?.

Figure 5: Household Budget Constraint

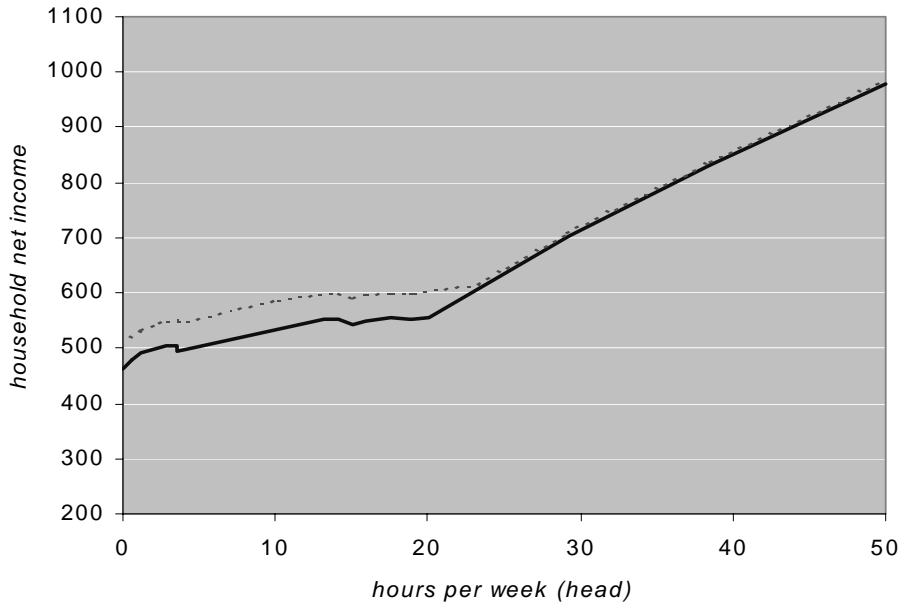


Figure 6: Marginal Effective Tax Rates

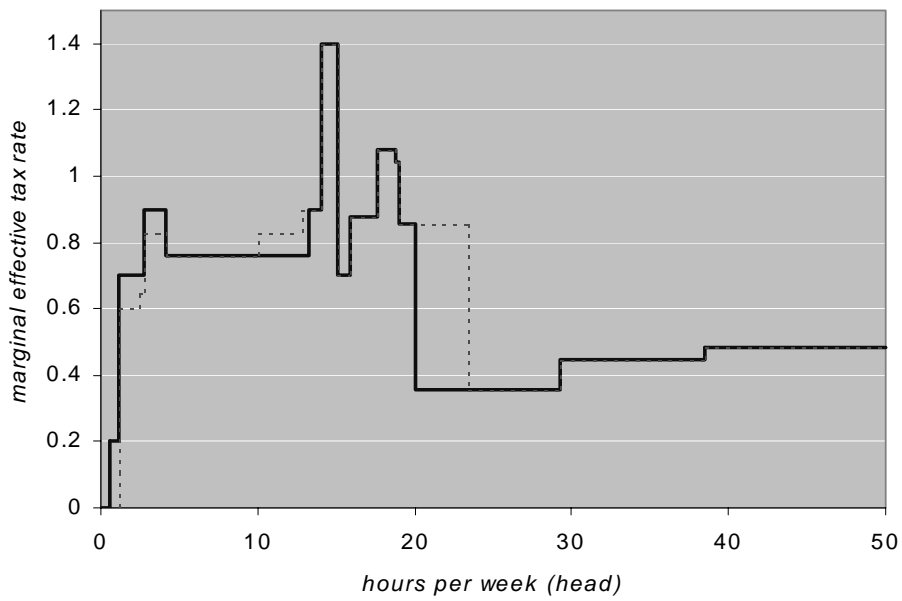


Table 1: METR diagnostics: first tax system

| hours | gross income | net income | 'virtual' income | METR  | probable reason for kink                       |
|-------|--------------|------------|------------------|-------|--|
| 0.00  | 0.00         | 532.43     | 532.43           | 0.000 | constraint starts                              |
| 0.58  | 14.45        | 546.87     | 535.31           | 0.200 | Income Tax starts                              |
| 1.20  | 30.00        | 559.32     | 550.31           | 0.700 | MR changes for Taxable Allowance               |
| 2.80  | 70.00        | 571.31     | 564.31           | 0.900 | MR changes for Beneficiary Tax Rebate          |
| 3.52  | 87.94        | 573.11     | 564.31           | .     | — discontinuity —                              |
| 3.52  | 87.97        | 563.49     | 554.70           | 0.900 | Family Tax Payment Pt B stops for partner      |
| 4.12  | 102.93       | 564.99     | 540.29           | 0.760 | Beneficiary Tax Rebate stops                   |
| 13.22 | 330.43       | 619.59     | 586.55           | 0.900 | Taxable Allowance stops for partner            |
| 14.08 | 351.91       | 621.72     | 762.49           | 1.400 | partner's MR changes for Non-Taxable Allowance |
| 15.10 | 377.60       | 611.47     | 498.19           | 0.700 | partner's MR changes for Family Payment        |
| 15.92 | 398.10       | 617.61     | 569.84           | 0.880 | Non-Taxable Allowance stops for partner        |
| 17.60 | 440.03       | 622.65     | 657.85           | 1.080 | MR changes for Low Income Rebate               |
| 18.81 | 470.22       | 620.23     | 639.04           | 1.040 | Medicare Levy starts                           |
| 19.03 | 475.71       | 620.01     | 551.03           | 0.855 | Low Income Rebate stops                        |
| 24.30 | 607.61       | 639.15     | 247.24           | 0.355 | MR changes for Medicare Levy                   |
| 29.23 | 730.77       | 718.59     | 313.01           | 0.445 | partner's MR changes for Family Payment        |
| 38.46 | 961.54       | 846.67     | 351.47           | 0.485 | Family Tax Payment Pt A stops for partner      |
| 50.00 | 1250.00      | 995.22     | 351.47           | .     | Family Tax Assistance starts                   |
|       |              |            |                  |       | MR changes for Income Tax                      |
|       |              |            |                  |       | MR changes for Income Tax                      |
|       |              |            |                  |       | constraint ends                                |

Notes: Budget constraints and METR diagnostics generated using the Melbourne Institute Tax and Transfer Simulator (MITTS), and arebased on the March 1999 Australian tax and transfer system. Budget sets and marginal effective tax rates details relate to the hours choices of the principle earner in a two-earner couple with two pre-school children and one school aged child. The secondary earner works for five hours per week, and both adults in the couple receive a gross hourly wage of Aus\$25. See ? for further details.

Table 2: METR diagnostics: second tax system

| hours | gross income | net income | 'virtual' income | METR  | probable reason for kink   |
|-------|--------------|------------|------------------|-------|--|
| 0.00  | 0.00         | 576.11     | 576.11           | 0.000 | constraint starts  |
| 1.15  | 28.87        | 604.97     | 578.99           | 0.100 | Income Tax starts  |
| 1.20  | 30.00        | 606.00     | 593.99           | 0.600 | MR changes for Taxable Allowance<br>MR changes for Beneficiary Tax Rebate  |
| 2.43  | 60.82        | 618.32     | 597.03           | 0.650 | (kink not recognised)  |
| 2.80  | 70.00        | 621.53     | 609.63           | 0.830 | MR changes for Taxable Allowance<br>MR changes for Beneficiary Tax Rebate  |
| 3.52  | 87.94        | 624.58     | 976.35           | .     | — discontinuity —  |
| 3.52  | 87.97        | 619.78     | 604.82           | 0.830 | Family Tax Payment Pt B stops for partner  |
| 4.12  | 102.96       | 622.33     | 597.62           | 0.760 | Beneficiary Tax Rebate stops   |
| 10.05 | 251.31       | 657.93     | 615.21           | 0.830 | Taxable Allowance stops  |
| 12.78 | 319.46       | 669.52     | 637.57           | 0.900 | Family Tax Payment Pt B stops for partner  |
| 14.08 | 351.88       | 672.76     | 813.51           | 1.400 | partner's MR changes for Family Payment  |
| 15.10 | 377.60       | 662.49     | 549.21           | 0.700 | Non-Taxable Allowance stops for partner  |
| 15.92 | 398.08       | 668.63     | 620.87           | 0.880 | MR changes for Low Income Rebate   |
| 17.60 | 440.05       | 673.67     | 708.87           | 1.080 | Medicare Levy starts   |
| 18.81 | 470.22       | 671.25     | 690.06           | 1.040 | Low Income Rebate stops  |
| 19.03 | 475.73       | 671.04     | 602.06           | 0.855 | MR changes for Medicare Levy   |
| 28.03 | 700.81       | 703.69     | 251.67           | 0.355 | partner's MR changes for Family Payment<br>Family Tax Payment Pt A stops for partner<br>Family Tax Assistance starts |
| 29.23 | 730.77       | 723.01     | 317.43           | 0.445 | MR changes for Income Tax  |
| 38.46 | 961.54       | 851.09     | 355.90           | 0.485 | MR changes for Income Tax  |
| 50.00 | 1250.00      | 999.65     | 355.90           | .     | constraint ends  |

Notes: as for Table 1. Relative to the March 1999 Australian tax and transfer system, this system involves a lower (10 per cent) tax rate for the first \$10,000 of the primary earner's income, and an increase to \$140 per fortnight in the basic rate of Family Payment.

Once each kink has been accurately identified, the reason (or reasons) for the presence of the turning point may be diagnosed by perturbing hours around the kink and examining the changes in the constituent parts of total household net income. In Tables 1 and 2 the turning points are rationalised for, respectively, the March 1999 Australian tax and transfer system and a variant of the same.<sup>16</sup> The advantage of the generic budget constraint algorithm over a grid search procedure is evident from these diagnostics. We are able to identify each turning point with high precision over the full budget constraint, and to explain which changes in entitlement or payment give rise to each turning point.

In applying the algorithm in the context of behavioural microsimulation, the position (number of hours) of all kinks and slopes of all linear sections of the budget constraint of each individual in a large and representative sample are automatically computed. In addition, the intersection of each linear section with the vertical axis (that is, net income corresponding to zero hours or gross earned income) is also computed as part of the algorithm. For each linear section, the slope is identified as the net wage rate and the intercept is referred to as the ‘virtual income’ corresponding to that section. The set of net wages, virtual incomes and hours at which the net wage changes (that is, the kink points) may then be retained for use in the calculation of average tax rates and replacement rates using the formulations described earlier. Figures 7 and 8 show, respectively, the average marginal tax rate and replacement rate schedules for the example household described earlier. These schedules are exact over the full range of hours, yet are based on evaluations from a tax microsimulation model only at each turning point in the hours range.

## 5 Conclusions

In this short paper we describe a generic method by which complete and accurate budget constraints may be generated from standard tax microsimulation models. The algorithm is appropriate for any microsimulation model which allows for variation in gross income (or hours of work) at the level of the individual. Our method takes advantage of recursions which exploit the geometric piecewise linear nature of the

---

<sup>16</sup>As a counterfactual to the March 1999 Australian tax and transfer system, we impose a lower (10 per cent) tax rate for the first \$10,000 of the primary earner’s income, and an increase to \$140 per fortnight in the basic rate of Family Payment.

Figure 7: Average Marginal Tax Rates

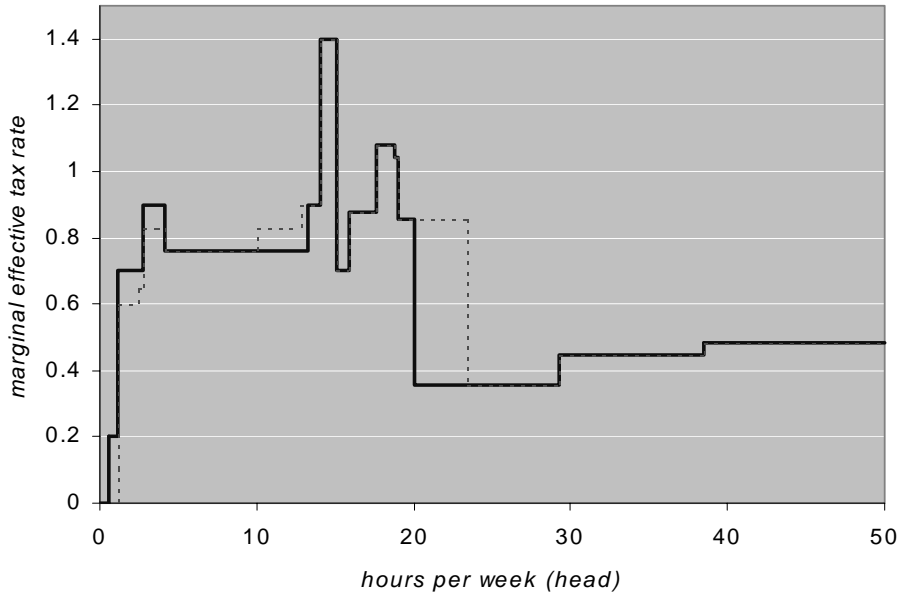
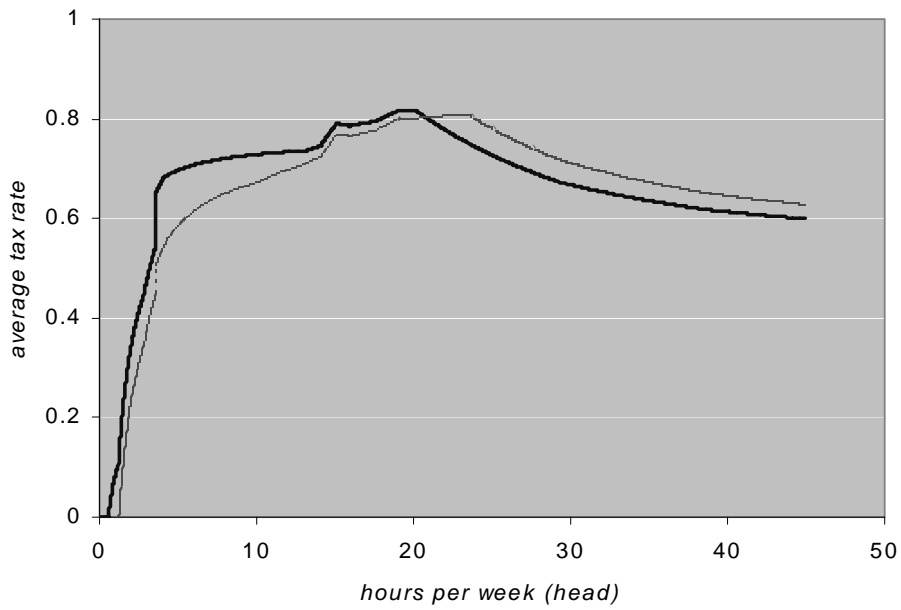


Figure 8: Replacement Rates



typical budget constraint. The result is an algorithm which is efficient, which identifies exactly the points on a budget constraint at which marginal effective tax rates change, and which can be adapted to generate continuous replacement rates and average tax rate schedules. We discuss how the algorithm may be adapted to produce fully continuous expressions for marginal effective tax rate (METR), average tax rate (ATR) and replacement rate (RR) measures at the level of the individual. We also discuss an extension to the basic algorithm which serves automatically to identify reasons for the change in marginal effective tax rates. A sketch of the algorithm code (in JAVA) is provided to illustrate the ease with which our methods may be implemented in practice.