A RECURSIVE ALGORITHM TO GENERATE PIECEWISE LINEAR BUDGET CONSTRAINTS

Alan Duncan
Graham Stark

THE INSTITUTE FOR FISCAL STUDIES
WP 00/11

A Recursive Algorithm to Generate Piecewise Linear Budget Constraints

Alan Duncan*and Graham Stark†

May 2000

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 derive 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.

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

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

*School of Economics, University of Nottingham, University Park, Nottingham NG7 2RD, United Kingdom and Institute for Fiscal Studies. Tel: +44 115 846 6107; e-mail: alan.duncan@nottingham.ac.uk

†Institute for Fiscal Studies, 7 Ridgmount Street, London WC1E 7AE, United Kingdom. Tel: +44 171 291 4800; e-mail: g_stark@ifs.org.uk

cost to the Government.[1] 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.[2] These simulations tend to be 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. At best, tax microsimulation models have sought to approximate full budget constraints by repeatedly evaluating net incomes at incremental gross income levels. This is an inefficient method which requires many unneccessary 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. 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 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. It all makes for some interesting budget constraints. Taxes and means-tested benefits can combine to produce marginal tax rates close to 100% (or even - though this is rarer than it used to be - in excess of 100%, so families *lose* income as they work more). There can be startling discontinuities as particular taxes or benefits kick in or are stopped

[1] For examples of the tax microsimulation approach, see *inter alia* Giles and McCrae (1995) for a UK application; Callan, O'Donoghue, and O'Neill (1996) for Ireland; Creedy and Duncan (1999) for Australia.

[2] Some countries also apply behavioural microsimulation methods to model employment incentive effects, although this is less common.

as earnings or hours worked exceed some level. With a system as compex as this, the availability of a generic algorithm to generate exact and authentic budget constraints is an extremely valuable tool.[3]

A complete budget constraint forms an input for the generation of *continuous* summary measures which give insight into the distortionary impact of the tax system, and the likely incentive impact of tax reform. It is often instructive to compare the impact of complete tax and transfer systems on a number of measures, including marginal effective tax rates (METRs), average tax rates (ATRs) and replacement rates (RRs) for a range of family types under different economic circumstances.

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. We also discuss an extension to the basic algorithm which serves automatically to identify reasons for the 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. In particular, our algorithm requires a tax microsimulation procedure of the form

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

[3]Indeed, tax systems are typically 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.

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 kth member of the tax unit. Net income is conditioned on gross wages $W = \{w_1, .., w_K\}$ for each person in the tax unit, hours of work H_{-k} for all people other than the kth 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.[4]

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;

2. takeup of benefits is complete, and taxes are universally paid. There is some evidence[5] to suggest that small amounts of benefits are less likely to be claimed.

[4] Discontinuities in the budget constraint obviously complicate the intuition, but may also be accommodated within the same generic algorithm.

[5] see Fry and Stark (1993)

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 timing issues. For example, National Insurance (NI) in the UK is recalulated weekly whilst the new Working Families' Tax Credit (WFTC) is paid for six months regardless of any change in circumstances.[6] Implicitly, our budget constraints show the long run position. However, 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, Δh, 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 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.

[6] See Blundell, Duncan, McCrae, and Meghir (2000).

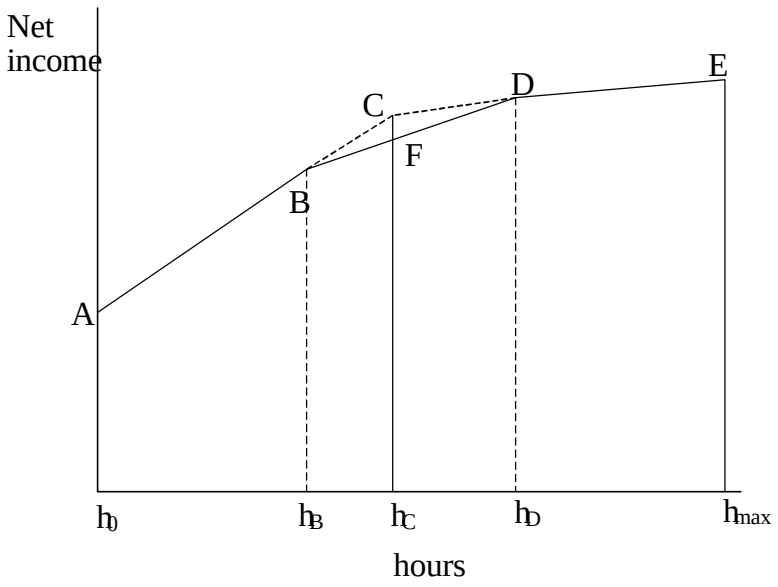


Figure 1:

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.[7]

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

[7] The procedure assumes that, where lines from two points are found to coincide, there are no intermediate kinks involved. This is because is it improbable that any practical budget constraint would take such a form.

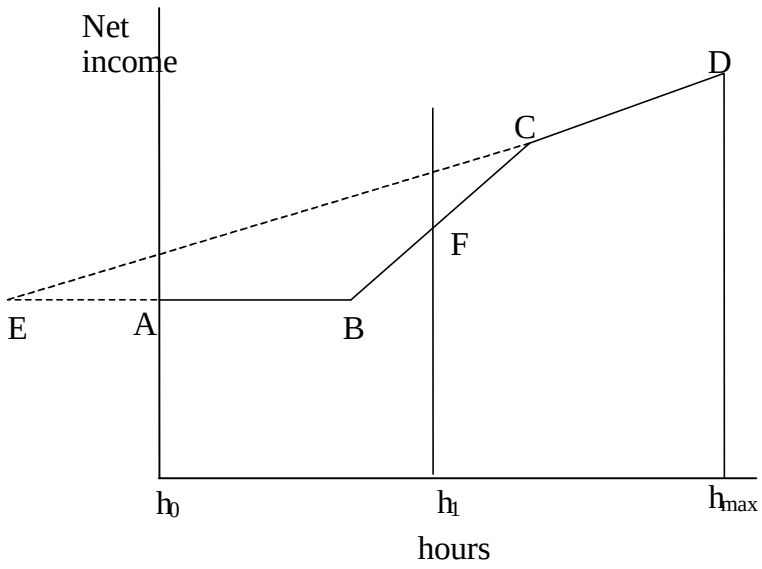


Figure 2:

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

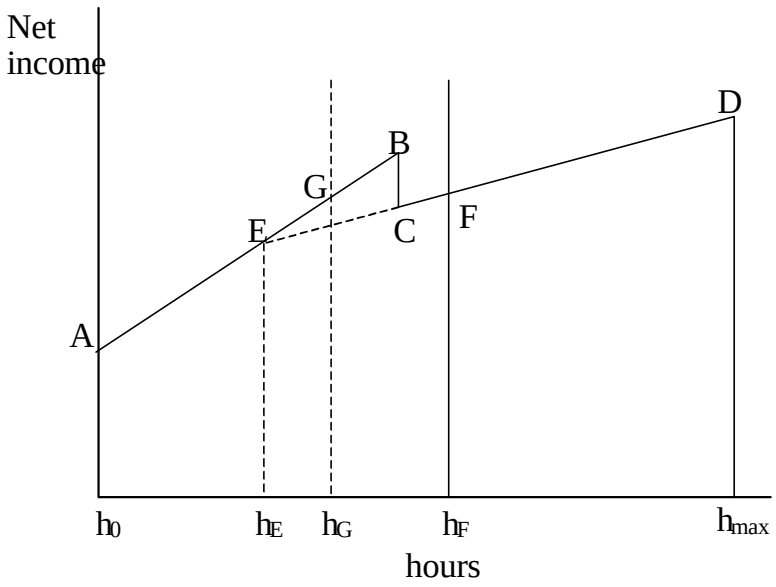


Figure 3:

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.

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.

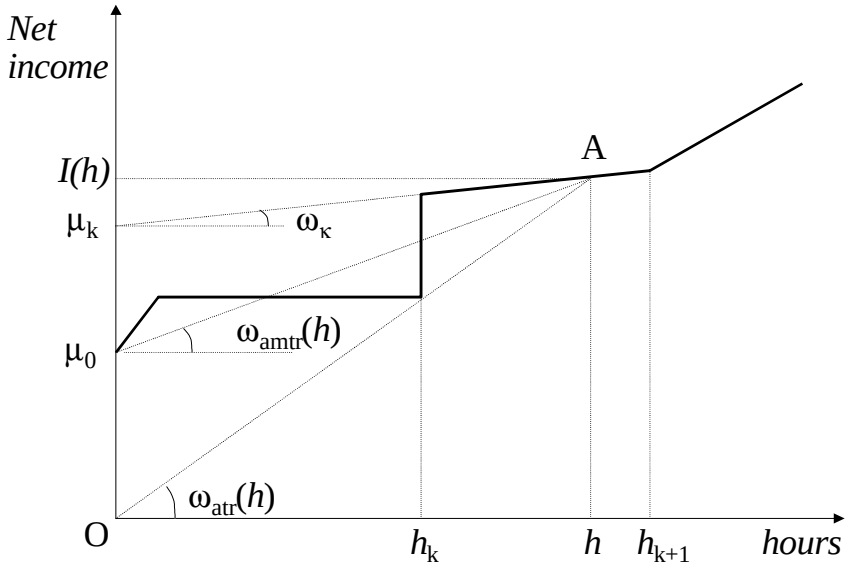
The general rules therefore apply of dividing the relevant range of hours into two sub-ranges whenever a fixed point is observed or whenever an intersection outside the relevant range is observed. Whenever an intersection point is found in the feasible range, the lines either side of that point are investigated. 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.

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' μ_k and marginal wage ω_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 μ_k and marginal wage rates ω_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, μ_k and ω_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;

Figure 4: Deriving Average Tax Rates and Replacement Rates



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 $AMTR(h)$.[8] (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 $\mu_0 A$, and intersects the kth linearised budget constraint segment μ_kA at net income $I(h)$, whereas the average wage $\omega_{atr}(h)$ is given by the slope OA.

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

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

where $AMTR_k(h)$ represents the average marginal tax rate at any point on the kth

[8] We thank John Creedy for pointing out this distinction.

linearised budget segment, and W denotes the gross wage rate. This may be re-arranged to give

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

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

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

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

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

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

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

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

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

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 μ_0 by total net income $I(h)$ at any point on the budget constraint (see Figure 4). However, since $I(h) = \mu_k + \omega_k.h$ 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

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

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

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

3.3 the marginal effective tax rate (METR) schedule

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

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

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 kth kink point on a budget constraint, and suppose that

net income $F(.)$ is evaluated at three successive points h_k-Δh , h_k and h_k+Δh for some small Δh. 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+Δh, 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-Δh 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-Δh and h_k is different to the increment between h_k and h_k+Δh, then the kink can be partially explained by a change in the withdrawal taper or tax rate.

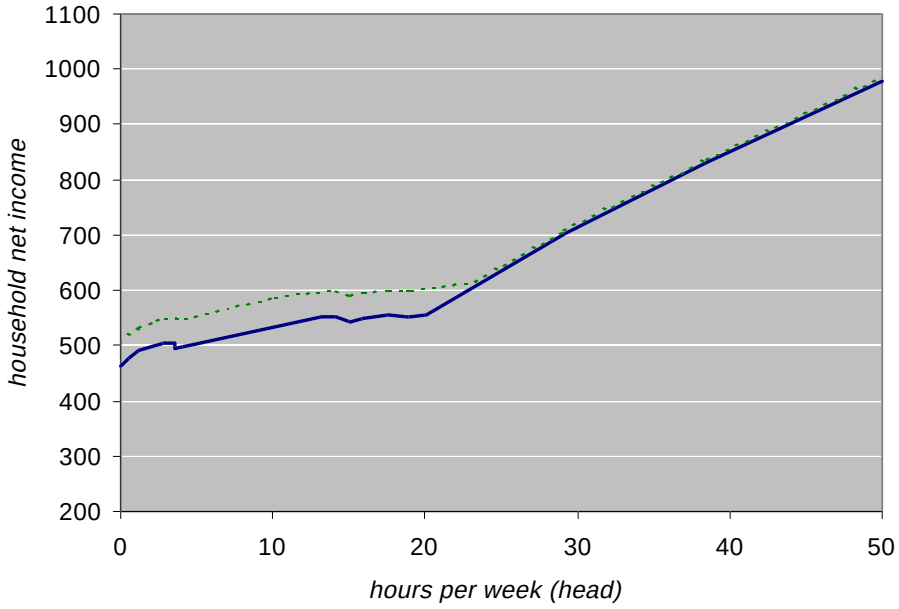
Of course, any one turning point in a budget constraint may be caused by more than one of the rules listed above. The present implementation requires a certain amount of local knowledge of the tax system, and so can't easily be generalised. It may also fail to identify explanations in all cases. Nevertheless, 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 Duncan-Stark algorithm by presenting budget contraints, 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), a new microsimulation model of the Australian tax and transfer system under development at the Melbourne Institute of Applied Economic and Social Research.[9] We focus on the budget contraint facing the primary earner in a two-person household under the March 1999 Australian tax and transfer system.

[9] For full details of the Melbourne Institute Tax and Transfer Simulator, see Creedy and Duncan (1999).

Figure 5: Household Budget Constraint



Figures 5 and 6 present graphical representations of the budget contraint 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.

Once each kink has been accurately identified, we proceed to diagnose the reason (or reasons) for the presence of the turning point 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.[10] 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.

[10] 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.

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
					MR changes for Beneficiary Tax Rebate
2.80	70.00	571.31	564.31	0.900	MR changes for Taxable Allowance
					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
					partner's MR changes for Non-Taxable Allowance
14.08	351.91	621.72	762.49	1.400	partner's MR changes for Family Payment
15.10	377.60	611.47	498.19	0.700	Non-Taxable Allowance stops for partner
15.92	398.10	617.61	569.84	0.880	MR changes for Low Income Rebate
17.60	440.03	622.65	657.85	1.080	Medicare Levy starts
18.81	470.22	620.23	639.04	1.040	Low Income Rebate stops
19.03	475.71	620.01	551.03	0.855	MR changes for Medicare Levy
24.30	607.61	639.15	247.24	0.355	partner's MR changes for Family Payment
					Family Tax Payment Pt A stops for partner
					Family Tax Assistance starts
29.23	730.77	718.59	313.01	0.445	MR changes for Income Tax
38.46	961.54	846.67	351.47	0.485	MR changes for Income Tax
50.00	1250.00	995.22	351.47	.	constraint ends

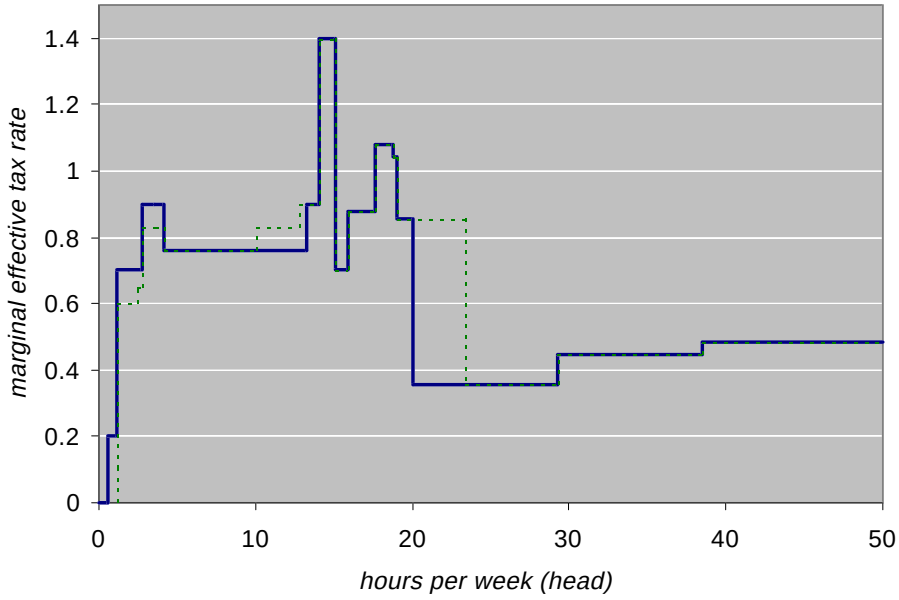
Notes: Budget constraints and METR diagnostics generated using the Melbourne Institute Tax and Transfer Simulator (MITTS), and arebased on the March 1999 Australian tax and transfer system. Budget sets and marginal effective tax rates details relate to the hours choices of the principle earner in a two-earner 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 Creedy and Duncan (1999) 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
					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
					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	(kink not recognised)
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	(kink not recognised)
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.

Figure 6: Marginal Effective Tax Rates



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, it is necessary to compute, for each linear segment, the intersection of that linear section when extended to the horizontal axis (that is, net income corresponding to zero hours). 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.

Figure 7: Average Marginal Tax Rates

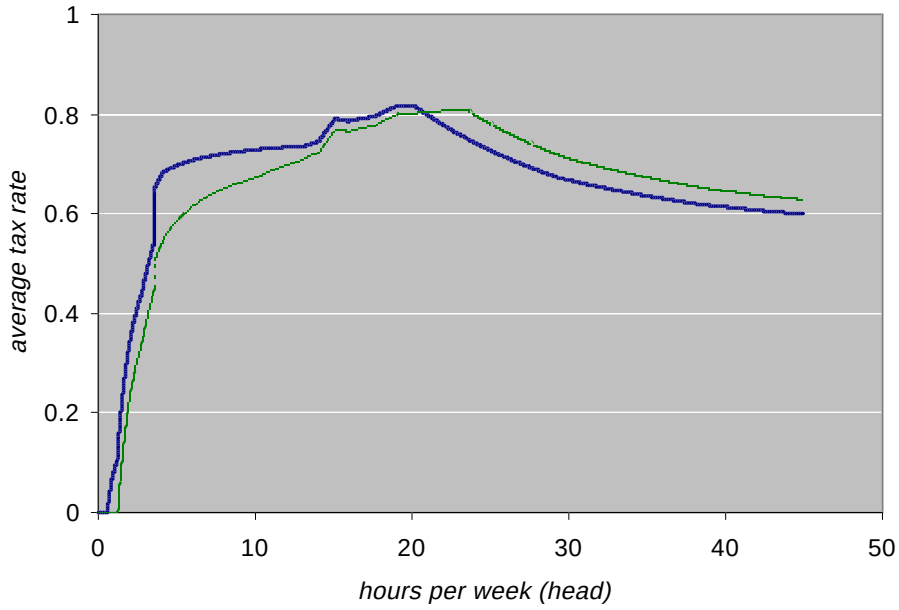
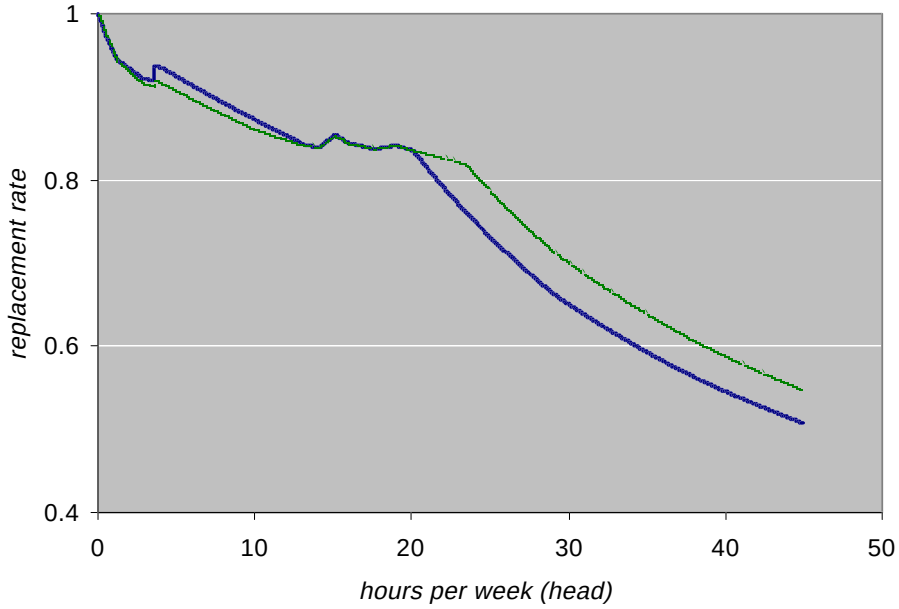


Figure 8: Replacement Rates



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 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 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 (in Java) is provided to

References

Blundell, R., A. Duncan, J. McCrae, and C. Meghir (2000): "The Labour Market Impact of the Working Families' Tax Credit," *Fiscal Studies*, 21, 75–104.

Callan, T., C. O'Donoghue, and C. O'Neill (1996): *Simulating Welfare and Income Tax Changes: The ESRI Tax Benefit Model SWITCH*. Dublin: ESRI.

Creedy, J., and A. Duncan (1999): "The Melbourne Institute Tax and Transfer Simulator," Discussion paper, Melbourne Institute, University of Melbourne.

Fry, V., and G. Stark (1993): "The Take-Up of Means-Tested Benefits, 1984-90," Discussion paper, London: Institute for Fiscal Studies.

Giles, C., and J. McCrae (1995): "The IFS Tax Micro-Simulation Tax and Benefit Model," Discussion Paper W95/19, Institute for Fiscal Studies.

Sedgewick, R. (1989): *Algorithms*. Reading, Mass.: Addison Wesley, 2nd edn.

6 Appendix: A Sketch of the Algorithm in JAVA

Our algorithm had been implemented in several computer languages: originally Modula-2 for the Institute for Fiscal Studies' tax microsimulation model TAXBEN[11], subsequently GAUSS for MITTS[12] and most recently JAVA for the Virtual Economy (VE) project. Here, we show a slightly edited version of the main routine from the JAVA version.[13]

All our implementations so far have used recursion - functions that call themselves. As is well known[14] 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 GOTOs statements at some cost to readability.[15]

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 contraint generator algorithm is then itself encapsulated in a class which takes the details of a particular household as input and returns the collection of (x, y) points which make up the budget constraint.

[11] Giles and McCrae (1995).

[12] Creedy and Duncan (1999).

[13] 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.

[14] See Sedgewick (1989) Chapter 5, pp51.

[15] See Sedgewick (1989) pp61 for an example.

Our implementation requires some control parameters:

- `incr`: the increment to add or subtract from each point;
- `start, stop`: the x-coordinate of the extreme points of the current segment

In the following JAVA implementation, these are contained in the `KinksControl` record. 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:

6.1 JAVA routine for generating budget constraints

```
1 public void GenerateBudgetConstraint( KinksControl k ) {
   2 double gross , z , x;
   3 Point[] p = new Point[ 5 ];
   4 Line[ ] line = new Line[ 2 ];
   5 gross = k.start;
   6 p[ 0 ] = F( gross );
   7 gross = gross + k.incr;
   8 p[ 1 ] = F( gross );
   9 gross = k.stop;
   10 p[ 2 ] = F( gross );
   11 gross = gross - k.incr;
   12 p[ 3 ] = F( gross );
   13 line[ 0 ] = new Line( p[ 0 ] , p[ 1 ] );
   14 line[ 1 ] = new Line( p[ 3 ] , p[ 4 ] );
   15 if line[ 1 ].sameLineAs( line[ 2 ] ){
   16     points.put( new Double( p[ 0 ].x ) , p[ 0 ] );
   17     points.put( new Double( p[ 3 ].x ) , p[ 3 ] );
```

```
18     return; }
19 else if (IMath.nearlyEqual( k.start , k.stop , k.incr / 100.0 ) )

20     return
21 else {
22     p[ 4 ] = line[ 0 ].intersection( line[ 1 ] );
23     if ( p[ 4 ].x <= k.start )
24         z = k.start + ( (k.stop - k.start ) / 2.0 );
25     else if (p[ 4 ] >= k.stop )
26         z = k.stop + ( (k.stop - k.start ) / 2.0 );
27     else
28         z = p[ 4 ].x;
29         x = k.stop;
30         k.stop = z;
31         GenerateBudgetConstraint( k );
32         k.start = z;
33         k.stop = x;
34         GenerateBudgetConstraint( k );
35     }
36 }
```

Here `F(gross)` is our call to the tax microsimulation procedure. This returns a point (x, y) where y is the tax unit net income at gross income x. The routine may equally well be defined in terms of hours of work h rather than gross income x, since $x = Wh$ for (fixed) gross wage W. At lines 5-14 this procedure is called four times to generate the lines corresponding to AC and CE in Figure 1.

Lines 15-18 test for whether these 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 19-20 are a simple overflow check for cases where the budget constraint is near-vertical.

If neither of these conditions are met, the algorithm continues (lines 22-34). Line 22 calculates point C in Figure 1. At lines 30-34 `GenerateBudgetConstraint` is called recursively to calculate the sub-budget constraints either side of C and so on until the

halting conditions at lines 15-20 are met. Finally, lines 23-26 are the non-convexity checks discussed in Section 2.2.

The initial call to `GenerateBudgetConstraint` would have `k.start` and `k.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.