

Integrating Harvest Cost Estimation into Woodstock

Gregory Paradis, ing.f., ing., M.Sc.E.

November 25, 2011

Abstract

An important part of integrating supply chain (ie. fiber demand) modelling with wood supply modelling is estimating net value creation potential of harvested timber, which involves subtracting estimated unit value of timber¹ from sum of cost components. An important cost component is harvest cost. The easiest way to model harvest cost in Woodstock² is to multiply harvested volume by a constant unit harvest cost coefficient, however this approach fails to capture variations in harvest cost (both within planning horizon for a given scenario, and across scenarios) attributable to variation in stand characteristics (age, stem density, stem size) of harvested stands. We have developed technology that allows integration of complex harvest cost performance indicators into Woodstock models.

1 Background

Unit cost of harvesting with a given system C_S can be estimated by summing unit cost of machines components in the harvesting system (eg. feller-buncher, skidder, delimber):

$$C_S = \sum_{m \in M} C_m$$

Unit cost of a given machine component C_m ³ can be estimated from product of machine productivity P_m ⁴, machine utilisation coefficient U_m ⁵, and machine rental rate R_m ⁶:

¹Value of timber must established relative to a given decoupling point (eg. sorted logs at roadside).

²Woodstock modelling software is distributed by Remsoft (www.remsoft.com).

³Measured in $\$/m^3$.

⁴Measured in m^3/PMH .

⁵Derived from ratio of productive machine hours (PMH) to scheduled machine hours (SMH).

⁶Measured in $\$/SMH$.

$$C_m = P_m \times U_m \times R_m$$

Machine utilisation coefficient U_m and machine rental rate R_m are typically modelled as using constant parameters based on machine operator efficiency and reliability, and case-specific nature of business relationship with machine owners. These constant parameters will not vary as a function of harvested stand attributes (ie. age, stem density, stem size, species composition) and harvesting intensity (ie. clearcut versus partial cut). In contrast, machine productivity P_m many vary significantly as a function of harvested stand attributes and harvesting intensity. We use machine productivity regressions published by MRNF as part of stumpage fee calculation model⁷ used for all harvesting on public land in Quebec.

Machine productivity regressions estimate machine as a function of stand age. There are different regressions for different types of machine. Harvesting machine productivity regressions use stem density (ie. stem count per unit area) and mean stem size (ie. units of timber volume per stem) as regression variables. Typically, these variables vary significantly as a function of stand age and stand type. Stand-wise stem density and stem size curves are not included in legacy Sylva II wood supply models, however Sylva II models use timber yield curve model from Pothier and Savard [1998]. Besides timber volume yield model, Pothier and Savard also publish stem density model using identical regression parameters as volume yield model (ie. IQS⁸ and IDR⁹) as volume yield model¹⁰. If we can determine regression parameters (IQS, IDR) used in Sylva II yield curves for a given stand, we can generate matching stem density curve set. Stem size curves can then easily be derived from quotient of volume and stem density curves. This module is coded in Python.

We have developed a module (“curve module”) that can reverse-engineer a Sylva II volume yield curve to deduce IQS and IDR parameters used to generate it, generate matching stem density curve using Pothier and Savard model, and derive stem volume curve. These curves can be output in a format that is easy to import into YIELDS section of a Woodstock model.

We have also developed a DLL extension module (“DLL module”) for Woodstock¹¹ that adds unit harvest cost curves to a model. Input for the DLL module includes stem density and stem volume curves. These curve can be generated using the curve module described earlier. Additionnal input for DLL module includes several constant coefficients (productivity function regression coefficients, machine rental rates, machine utilisation coefficients, mean skidding distances, mean skid load size, calibration factors, etc). These parameters can be defined in the CONSTANTS section of a Woodstock model.

⁷<http://www.mrnf.gouv.qc.ca/forets/entreprises/entreprises-tarification.jsp>

⁸Indice de Qualité de Station

⁹Indice de Densité Relative

¹⁰Thanks to Pascal Gauthier for this hint.

¹¹<http://www.remsoft.com/docs/library/dll.pdf>

2 Curve Module

Curve module includes curve-matching functions that can be used to reverse-engineer IQS and IDR parameters for a given Pothier-derived species-wise yield curve. Curve module also includes functions for generating stem density curves for any combination of IQS, IDR and species, using regression functions published in Pothier and Savard [1998]. Mean stem size curves can easily be derived from volume yield and stem density curves.

You will have to code an appropriate main program loop to calling the functions in this module.

3 DLL Module

To use the DLL extension module with your Woodstock model:

1. Copy the DLL file to the Woodstock model root directory (ie. directory containing PRI file).
2. Add appropriate parameter definitions to the *CONSTANTS* section.
3. Map parameter definitions to forest types in *LANDSCAPE* section.
4. Add stem density and stem size curves to the *YIELDS* section.
5. Add harvest cost curves to the *YIELDS* section (using *_DLL* function).
6. Add harvest cost outputs to the *OUTPUTS* section.

Examples of appropriate Woodstock code are provided below.

3.1 CONSTANTS Section

Sample code adding appropriate parameter definitions which can be used as input parameters for harvest cost DLL extension module:

```
dmbcoef0 1
dmbcoef1 89.37
dmbcoef2 80.83
fellcoef1 57.42
fellcoef2 54.55
fellcoef3 41.33
slshcoef1 38.83
slshcoef2 28.78
slshcoef3 19.80
regcalib1 1.00
regcalib2 1.00
regcalib3 1.00
rentalrate0 0
rentalrate1 170
rentalrate2 120
rentalrate3 110
rentalrate4 90
rentalrate5 35
skiddist1 350
skidload1 2.5
```

The following table describes these parameters.

Parameter	Description	Note
dlmbcoef0	Delimber productivity function regression parameter (hardwood)	Do not modify
dlmbcoef1	Delimber productivity function regression parameter (mixedwood)	Do not modify
dlmbcoef2	Delimber productivity function regression parameter (softwood)	Do not modify
fellcoef1	Feller-buncher productivity function regression parameter (softwood)	Do not modify
fellcoef2	Feller-buncher productivity function regression parameter (hardwood, mixedwood)	Do not modify
fellcoef3	Feller-buncher productivity function regression parameter (cover type?)	Do not modify
slshcoef1	Slasher productivity function regression parameter (hardwood)	Do not modify
slshcoef2	Slasher productivity function regression parameter (mixedwood)	Do not modify
slshcoef3	Slasher productivity function regression parameter (softwood)	Do not modify
regcalib1	Regression calibration coefficient (hardwood)	
regcalib2	Regression calibration coefficient (mixedwood)	
regcalib3	Regression calibration coefficient (softwood)	
rentalrate0	Delimber rental rate (hardwood), topper rental rate (mixedwood, softwood)	
rentalrate1	Feller-buncher rental rate	
rentalrate2	Skidder rental rate	
rentalrate3	Delimber rental rate	
rentalrate4	Slasher rental rate	
rentalrate5	Topper rental rate (motor manual chainsaw operator)	
skiddist1	Skidding distance (m)	
skidload1	Skidder load size (m^3)	

3.2 LANDSCAPE Section

Sample code adding mapping constant parameter values to different forest types:

```
*THEME cover_type
  f _INDEX(skiddist=#skiddist1,
           skidload=#skidload1,
           fellcoef=#fellcoef2,
```

```

        dlmbcoef=#dlmbcoef0,
        slshcoef=#slshcoef3,
        fellrate=#rentalrate1,
        skidrate=#rentalrate2,
        dlmbrate=#rentalrate0,
        slshrate=#rentalrate4,
        toprrate=#rentalrate5,
        regcalib=#regcalib3)
m _INDEX(skiddist=#skiddist1,
        skidload=#skidload1,
        fellcoef=#fellcoef2,
        dlmbcoef=#dlmbcoef2,
        slshcoef=#slshcoef2,
        fellrate=#rentalrate1,
        skidrate=#rentalrate2,
        dlmbrate=#rentalrate3,
        slshrate=#rentalrate4,
        toprrate=#rentalrate0,
        regcalib=#regcalib2)
r _INDEX(skiddist=#skiddist1,
        skidload=#skidload1,
        fellcoef=#fellcoef1,
        dlmbcoef=#dlmbcoef1,
        slshcoef=#slshcoef1,
        fellrate=#rentalrate1,
        skidrate=#rentalrate2,
        dlmbrate=#rentalrate3,
        slshrate=#rentalrate4,
        toprrate=#rentalrate0,
        regcalib=#regcalib1)

```

3.3 YIELDS Section

Sample code adding stem density and stem size yield curves which can be used as input parameters for harvest cost DLL extension module:

```

*Y ? ? ? ? ? f
  ystemsize 1 0.081 0.108 0.135 0.162 0.189 0.216 0.243 0.270 0.297 0.324
            0.351 0.378 0.405 0.432 0.459 0.486 0.513 0.54 0.567 0.594 0.621 0.648
            0.675 0.702 0.729 0.756 0.783 0.810 0.837 0.864 0.891 0.918 0.945
            0.972 0.999 1.026 1.053 1.080 1.107 1.134
  ystemdens 1 578 551 535 525 517 510 505 500 496 492 489 486 483 481 478 476
            474 472 471 469 467 466 464 463 462 460 459 458 457 456 455 454 453
            452 451 450 449 448 447 447
*Y ? ? ? ? ? m
  ystemsize 1 0.016 0.048 0.080 0.112 0.144 0.176 0.208 0.240 0.272 0.304
            0.336 0.368 0.400 0.432 0.464 0.496 0.528 0.560 0.592 0.624 0.656
            0.688 0.720 0.752 0.784 0.816 0.848 0.880 0.912 0.944 0.976 1.008
            1.040 1.072 1.104 1.136 1.168 1.200 1.232 1.264
  ystemdens 1 839 577 464 397 352 319 294 273 256 242 230 219 210 202 195 188
            182 176 171 167 162 158 154 151 148 145 142 139 136 134 131 129 127
            125 123 121 119 118 116 115
*Y ? ? ? ? ? r
  ystemsize 1 0.007 0.034 0.061 0.088 0.115 0.142 0.169 0.196 0.223 0.250
            0.277 0.304 0.331 0.358 0.385 0.412 0.439 0.466 0.493 0.520 0.547
            0.574 0.601 0.628 0.655 0.682 0.709 0.736 0.763 0.790 0.817 0.844
            0.871 0.898 0.925 0.952 0.979 1.006 1.033 1.060
  ystemdens 1 949 695 579 509 460 424 395 372 353 337 323 310 299 289 281 273
            265 258 252 246 241 236 231 227 223 219 215 212 209 205 202 200 197
            194 192 189 187

```

Sample code adding stem harvest cost curves using `_DLL` function:

```

*YC ? ? ? ? r ? ?
yhcostr _DLL(_VARIANT,_STDCALL) woodstockhcost.dll hcostr(ystemsize,
                                                         ystemdens,

```

```

yvol_total,
skiddist,
skidload,
fellcoef,
dlmbcoef,
slshcoef,
fellrate,
skidrate,
dlmbrate,
slshrate,
toprrate,
regcalib)

```

The Woodstock code above passes 14 parameters to hcostr function of DLL module:

```

void __stdcall hcostr(ParameterArray* params, SingleArray*& result) {
    //////////////////////////////////////
    // DO NOT EDIT THE FOLLOWING LINE //////////////////////////////////////
    for(int i=result->lowIndex; i<=result->highIndex; i++) result->data[i] = 0.0;

    //////////////////////////////////////
    // check parameter count and data types
    const int num_params = 14;
    if (params->count != num_params) exit(1); // wrong number of parameters
    char datatype[num_params] = {'A','A','A','S','S','S','S','S','S','S','S','S','S','S'};
    for (int i=0; i<num_params; i++)
        if (params->dataType[i] != datatype[i]) exit(1); // bad param
    // extract parameters
    SingleArray* size_curve = fill_front(static_cast<SingleArray*>(params->data[0]));
    SingleArray* dens_curve = fill_front(static_cast<SingleArray*>(params->data[1]));
    SingleArray* hvol_curve = fill_front(static_cast<SingleArray*>(params->data[2]));
    float skid_dist = (*static_cast<float*>(params->data[3]));
    float skid_load = (*static_cast<float*>(params->data[4]));
    float fell_coef = (*static_cast<float*>(params->data[5]));
    float dlmb_coef = (*static_cast<float*>(params->data[6]));
    float slsh_coef = (*static_cast<float*>(params->data[7]));
    float fell_rate = (*static_cast<float*>(params->data[8]));
    float skid_rate = (*static_cast<float*>(params->data[9]));
    float dlmb_rate = (*static_cast<float*>(params->data[10]));
    float slsh_rate = (*static_cast<float*>(params->data[11]));
    float topr_rate = (*static_cast<float*>(params->data[12]));
    float reg_calib = (*static_cast<float*>(params->data[13]));
    int tmpindex;
    for (int i=result->lowIndex+1; i<=result->highIndex; i++) {
        float slsh_size_exp = 0.26f;
        float fell_size_exp = 0.86f;
        float fell_dens_exp = 0.18f;
        float dlmb_size_exp = 0.68f;
        tmpindex = i;
        if (i>size_curve->highIndex) tmpindex = size_curve->highIndex;
        float size = static_cast<float>(size_curve->data[tmpindex]);
        tmpindex = i;
        if (i>dens_curve->highIndex) tmpindex = dens_curve->highIndex;
        float dens = static_cast<float>(dens_curve->data[tmpindex]);
        tmpindex = i;
        if (i>hvol_curve->highIndex) tmpindex = hvol_curve->highIndex;
        float hvol = static_cast<float>(hvol_curve->data[tmpindex]);
        result->data[i] += (slsh_rate/(slsh_coef*pow(size,slsh_size_exp)));
        result->data[i] += (size==0 || dens==0) ? 0 : (fell_rate/(fell_coef*pow(size,fell_size_exp)*pow(dens,fell_dens_exp)));
        result->data[i] += (skid_rate*(1.3f+0.0145f*skid_dist)/(54.0f*skid_load));
        result->data[i] += (size==0) ? 0 : (dlmb_rate/(dlmb_coef*pow(size,dlmb_size_exp)));
        result->data[i] *= reg_calib;
    }
}

```

3.4 OUTPUTS Section

Sample code adding harvest cost outputs by treatment type.

```
-----  
; temp outputs (used for harvest cost below)  
*OUTPUT ohcosttmp_tr-cprs temp output for harvest cost calculation  
  *SOURCE cprs yhcostr  
*OUTPUT ohcosttmp_tr-cprsbop95 temp output for harvest cost calculation  
  *SOURCE cprsbop95 yhcostr  
*OUTPUT ohcosttmp_tr-cprsthо temp output for harvest cost calculation  
  *SOURCE cprsthо yhcostr  
*OUTPUT ohcosttmp_tr-cprscpe temp output for harvest cost calculation  
  *SOURCE cprscpe yhcostr  
-----  
; harvest cost  
*OUTPUT ohcost_tr-cprs Harvest cost (treatment: cprs)  
  *SOURCE ohcosttmp_tr-cprs * ohvol_tr-cprs / otreatedarea_tr-cprs  
*OUTPUT ohcost_tr-cprsbop95 Harvest cost (treatment: cprsbop95)  
  *SOURCE ohcosttmp_tr-cprsbop95 * ohvol_tr-cprsbop95 / otreatedarea_tr-  
    cprsbop95  
*OUTPUT ohcost_tr-cprsthо Harvest cost (treatment: cprsthо)  
  *SOURCE ohcosttmp_tr-cprsthо * ohvol_tr-cprsthо / otreatedarea_tr-cprsthо  
*OUTPUT ohcost_tr-cprscpe Harvest cost (treatment: cprscpe)  
  *SOURCE ohcosttmp_tr-cprscpe * ohvol_tr-cprscpe / otreatedarea_tr-cprscpe
```

References

D. Pothier and F. Savard. *Actualisation des tables de production pour les principales espèces forestières du Québec*. Ministère des Ressources naturelles, 1998.