Modell över Sveriges primärenergitillförsel
Kurs ENM155

Andreas Hagesjö Daniel Pettersson Magnus Hagmar
Niclas Ogeryd Robert Nyquist

24 november 2014

1 Introduktion

Denna rapport innehåller en enkel modell utav Sveriges energisystem som det ser ut idag. Modellen visar hur olika primäreneriger födelas på de tre sektorerna industri, transport och bostäder samt en uppskattning utav Sveriges totala primärenergitillförsel.

2 Metod

Modellen är nedbruten i tre delar, industri, transport och bostäder som är de olika sektorerna. För varje sektor så listas alla primärenergier som bidrar till respektive sektor. Varje primärenergi går sedan vidare till de olika sekundärenergierna som den bidrar till. Varje sekundärenergi går vidare till sektorn, alternativ till en ny sekundärenergi som i sin tur går vidare till en sektor eller ytterligare en sekundärenergi.

- Då vi har brutit ner modellen i sektorer så följer de inte diagrammet i Figur 1 i lab PM. Istället så ger flödesschemat i Appendix A en ungefärlig överskådlig bild utav strukturen på vår implementation. Den egentliga anledningen till att vi skapade flödesschemat var för att kunna förstå och diskutera problemet i grupp.
- Modellen är byggd så att det går att ta reda på tillförseln av varje enskild primärenergi samt vilka typer av primärenergi, och mängden, varje enskild sektor använder. Det går även att räkna ut värden på sekundärenergierna för varje sektor med hjälp av modellen.
- Då varje sektor innehåller alla primärenergier och sekundärenergier som bidrar så blir det väldigt enkelt att addera nya energier. Den nya energin läggs till i sektorn den bidrar till och går sedan vidare till en sekundärenerig eller sektorn.

2.1 Matematisk modell

Vi definerar ett utryck för att beräkna elen i transport samt industrisektorn, som vi sedan referenserar för att minska längden på det totala uttrycket.

I våra ekvationer betyder E_{fe} energin från fossila bränslen omvandlat till el. Samma sätt med E_{fjv}, fjärrvärme till värme, E_{ffj}, fossil till fjärvärme etc.

2.2 Bostäder

$$EL = \frac{\frac{E_v k_{ev}}{\varphi_{ev}} + \frac{E_v k_{fjv} k_{vp}}{\varphi_{fjv}\varphi_{vp}\varphi_{trans}} + E_e}{\varphi_e \varphi_{trans}}$$

Fossila bränslen

$$E_f = \frac{E_v k_{fv}}{\varphi_{fv}} + \frac{E_v k_{fjv} k_{ffj}}{\varphi_{fjv}\varphi_{ffj}\varphi_{trans}} + EL \cdot \frac{k_{fe}}{\varphi_{fe}}$$

Biobränslen

$$E_b = \frac{E_v k_{bv}}{\varphi_{bv}} + \frac{E_v k_{fjv} k_{bfj}}{\varphi_{fjv}\varphi_{bfj}\varphi_{trans}} + EL \cdot \frac{k_{be}}{\varphi_{be}}$$

Vindkraft

$$E_{vind} = EL \cdot \frac{k_{vind}}{\varphi_{vind}}$$

Vattenkraft

$$E_{vatten} = EL \cdot \frac{k_{vatten}}{\varphi_{vatten}}$$

Kärnkraft

$$E_{karn} = EL \cdot \frac{k_{karn}}{\varphi_{karn}}$$

Spillvärme (inte en ”riktig” energi, men måste tas med i beräkningarna)

$$E_{spill} = EL \cdot \frac{k_{spill}}{\varphi_{spill}}$$

2.3 Transport

$$E_f = \frac{E_t k_f}{\varphi_f \varphi_{drivmedel}}$$
$$E_b = \frac{E_t k_b}{\varphi_b \varphi_{drivmedel}}$$

2.4 Industri

Industrisektorn skiljer sig från bostadssektorn genom att det inte finns någon eluppvärmning, annars är de två sektorerna lika.

$$EL = \frac{\frac{E_v k_{ev}}{\varphi_{ev}} + E_e}{\varphi_e \varphi_{trans}}$$

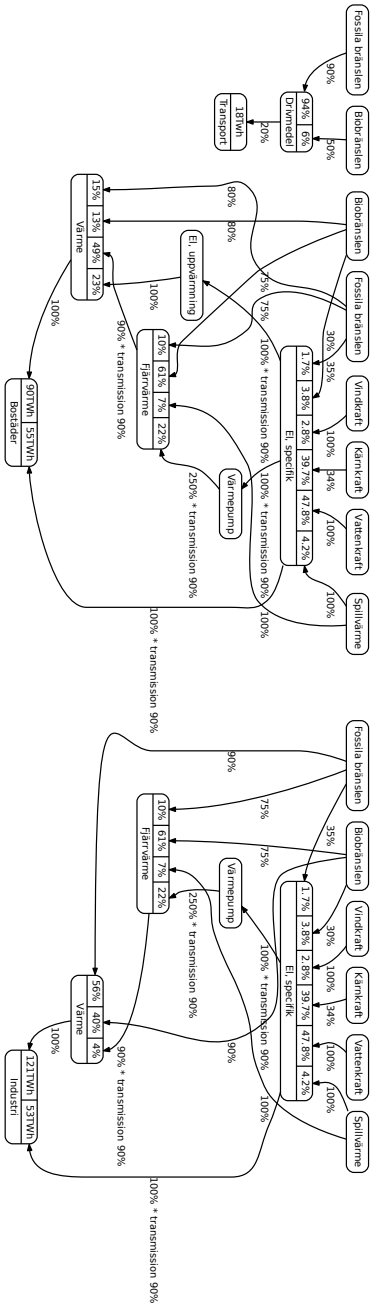
3 Resultat

I tabell 1 visas den totala energitillförseln samt varje enskild energikällas tillförsel.

Energikälla	Tillförsel
Fossila bränslen	201.483 TWh
Biobränsle	147.288 TWh
Vindkraft	4.187 TWh
Vattenkraft	71.491 TWh
Kärnkraft	174.638 TWh
Totalt	649.561 TWh

Tabell 1: Resultat

A Flödesschema



Figur 1: Flödesschema över energianvändning i Sverige.

B Programkod

I denna fil finns klasserna för sektorer samt energier.

```
class Sector:
  def __init__(self, id, name):
    self.name = name
    self.id = id
    self.energy = 0
    self.energies = {}

  def add_energy(self, id, energy):
    self.energies[id] = energy

  def value(self):
    return self.energy

class Energy:
  def __init__(self, id, name, energy=0):
    self.name = name
    self.id = id
    self.energy = energy
    self.sectors = {}
    self.inputs = {}
    self.subenergies = {}

  def add_input(self, id, energy, efficiency, quota):
    self.inputs[id] = (energy, efficiency, quota)

  def add_subenergy(self, id, subenergy, efficiency,
     quota):
    self.subenergies[id] = (subenergy, efficiency,
       quota)

  def add_sector(self, id, sector, efficiency, amount)
     :
    self.sectors[id] = (sector, efficiency, amount)

  def value(self, id=None):
    if id:
      (sum_used, sum_created) = self.sum_value_energy(
         self, id)
      return (sum_used, sum_created)
    else:
```

```
    (sum_used, sum_created) = self.sum_value_energy(
        self, 'all')
    return (sum_used, sum_created)

def sum_value_energy(self, energy, id):
  sum_used = 0
  sum_created = 0
  for subenergy_id in energy.subenergies:
    link = energy.subenergies[subenergy_id]
    if subenergy_id == id:
      created_temp = link[0].energy * link[2]
      sum_created += created_temp
      sum_used += created_temp / link[1]

    else:
      (used_temp, created_temp) = self.
          sum_value_energy(link[0], id)
      created_temp = created_temp * link[2]
      used_temp = used_temp * link[2] / link[1]
      sum_created += created_temp
      sum_used += used_temp

  (sum_used, sum_created) = self.sum_value_sector(
      energy, id, sum_used, sum_created)

  return (sum_used, sum_created)

def sum_value_sector(self, energy, id, sum_used,
    sum_created):
  for sector_id in energy.sectors:
    link = energy.sectors[sector_id]
    if sector_id == id or id == 'all':
      created_temp = link[2]
      sum_created += created_temp
      sum_used += created_temp / link[1]
  return (sum_used, sum_created)
```

Här är själva programmet.

```
#-*- encoding: utf-8 -*-
from Energy import Sector, Energy
from sys import argv, exit
from copy import copy
import json
import argparse

def main():
  parser = argparse.ArgumentParser(description=u"
      Modellering av Sveriges energiförbrukning")
  parser.add_argument("-t", "--total",action="
      store_true", dest="total", help=u"Visa totala
      energiförbrukningen för Sverige")
  parser.add_argument("-s", "--sectors",action="
      store_true", dest="sectors", help=u"Visa alla
      sektorer")
  parser.add_argument("-p", "--primary-energies",
      action="store_true", dest="primary", help=u"Visa
      alla primära energier")
  parser.add_argument("-e", "--energies",action="
      store_true", dest="energies", help=u"Visa alla
      energier")
  parser.add_argument("-v", "--value", metavar=("
      from_id", "to_id"),dest="values", type=str,nargs=
      '*', help=u"Visa hur mycket energi utav
      energitypen 'from_id' som används till
      energitypen eller sektorn 'to_id'. Visar även hur
      mycket energi man får ut i sekundärenergin eller
      sektorn efter alla energiomvandlingar och fö
      rluster. Anger man inte 'to_id' så tolkas detta
      som alla sektorer")
  args = parser.parse_args()

  with open("system-data.json", "r") as fp:
    obj = json.load(fp)

  (primaryenergies, energies, sectors) = build_model(
      obj)
  id_to_name = {}
  for e in energies:
    id_to_name[energies[e].id] = energies[e].name
  for s in sectors:
```

```
  id_to_name[sectors[s].id] = sectors[s].name

calculate_energies(energies, sectors)

if args.total:
  total = 0
  print(u"Sveriges energiförbrukning: ")
  for e in primaryenergies.values():
    total += e.energy
    print(u'{:20}{:10.3f} TWh'.format(e.name, e.
      energy))
  print(u"\n{:20}{:10.3f} TWh".format("Total energi"
    , total))
elif args.sectors:
  print('\n'.join([s.name for s in sectors.values()
    ]))
elif args.primary:
  print('\n'.join([p.name for p in primaryenergies.
    values()]))
elif args.energies:
  for e in energies.values():
    print u"{:16} id: {:15}".format(e.name, e.id)
elif args.values:
  length = (len(args.values))
  if length == 1:
    (used, created) = energies[args.values[0]].value
      ()
    output = u"{:0.3f} THw av energin från {:s} går
      till alla sektorer.".format(used, energies[
      args.values[0]].name)
    output2= u"Med detta så får man ut {:0.3f} TWh
      till alla sektorer.".format(created)
  if length == 2:
    (used, created) = energies[args.values[0]].value
      (args.values[1])
    output = u"{:0.3f} THw av energin från {:s} går
      till {:s}.".format(used, energies[args.values
      [0]].name, id_to_name[args.values[1]])
    output2= u"Med detta så får man ut {:0.3f} TWh
      till {:s}.".format(created, id_to_name[args.
      values[1]])
  if used==0:
    print("Parametrarna gav inget resultat")
    exit()
```

```
    print output
    print output2
  else:
    parser.print_help()

  return primaryenergies, energies, sectors

def build_model(obj):
  sectors = {}
  energies = {}
  primaryenergies = {}

  for energy_id in obj["primary_energies"]:
    energy_obj = obj["primary_energies"][energy_id]
    energy = Energy(energy_id, energy_obj["name"])
    primaryenergies[energy_id] = energy
    energies[energy_id] = energy

    add_inputs(obj, energy_id, energy_obj, energies)

  for energy_id in obj["energies"]:
    energy_obj = obj["energies"][energy_id]
    if not energy_id in energies:
      energies[energy_id] = Energy(energy_id, obj["
        energies"][energy_id]["name"])
    add_inputs(obj, energy_id, energy_obj, energies)
    add_sectors(obj, energy_id, energy_obj, sectors,
      energies)

  return (primaryenergies, energies, sectors)


def add_inputs(obj, id, energy_obj, energies):
  if "energies" in energy_obj:
    for energy_id in energy_obj["energies"]:
      if not energy_id in energies:
        energies[energy_id] = Energy(energy_id, obj["
          energies"][energy_id]["name"])
      efficiency = energy_obj["energies"][energy_id]["
        efficiency"]
      quota = energy_obj["energies"][energy_id]["quota
        "]
      energies[energy_id].add_input(id, energies[id],
        efficiency, quota)
```

```
        energies[id].add_subenergy(energy_id, energies[
            energy_id], efficiency, quota)

def add_sectors(obj, id, energy_obj, sectors, energies
    ):
  if "sectors" in energy_obj:
    for sector_id in energy_obj["sectors"]:
      if not sector_id in sectors:
        sectors[sector_id] = Sector(sector_id, obj["
            sectors"][sector_id]["name"])
      efficiency = energy_obj["sectors"][sector_id]["
          efficiency"]
      amount = energy_obj["sectors"][sector_id]["
          amount"]
      sectors[sector_id].add_energy(id, energies[id])
      sectors[sector_id].energy += amount
      energies[id].add_sector(sector_id, sectors[
          sector_id], efficiency, amount)

def calculate_energies(energies, sectors):
  for sector_id in sectors:
    sector = sectors[sector_id]
    for energy_id in sector.energies:
      energy = energies[energy_id]
      energy_sector_link = energy.sectors[sector_id]
      amount = energy_sector_link[2]/
          energy_sector_link[1]
      increase_energy(energy, amount, energies)

def increase_energy(energy, amount, energies):
  energy.energy += amount
  for input_id in energy.inputs:
    input = energies[input_id]
    link = energy.inputs[input.id]
    efficiency = link[1]
    amount_temp = amount * link[2] / efficiency
    increase_energy(input, amount_temp, energies)

if __name__ == "__main__":
  main()
```

Och här är inputdatan till vårt program.

```
{
  "name": "Sveriges primärenergitillförsel",
  "primary_energies": {
    "fossil": {
      "name": "Fossila bränslen",
      "energies": {
        "fuel": {
          "efficiency": 0.90,
          "quota": 0.94
        },
        "heating_residences": {
          "efficiency": 0.80,
          "quota": 0.15
        },
        "heating_industry": {
          "efficiency": 0.90,
          "quota": 0.56
        },
        "district_heating": {
          "efficiency": 0.75,
          "quota": 0.10
        },
        "electric": {
          "efficiency": 0.35,
          "quota": 0.017
        }
      }
    },
    "bio": {
      "name": "Biobränslen",
      "energies": {
        "fuel": {
          "efficiency": 0.50,
          "quota": 0.06
        },
        "heating_residences": {
          "efficiency": 0.80,
          "quota": 0.13
        },
        "heating_industry": {
          "efficiency": 0.90,
          "quota": 0.40
```

```
      },
      "district_heating": {
        "efficiency": 0.75,
        "quota": 0.61
      },
      "electric": {
        "efficiency": 0.30,
        "quota": 0.038
      }
    }
  },
  "nuclear": {
    "name": "Kärnkraft",
    "energies": {
      "electric": {
        "efficiency": 0.34,
        "quota": 0.397
      }
    }
  },
  "water": {
    "name": "Vattenkraft",
    "energies": {
      "electric": {
        "efficiency": 1.00,
        "quota": 0.478
      }
    }
  },
  "wind": {
    "name": "Vindkraft",
    "energies": {
      "electric": {
        "efficiency": 1.00,
        "quota": 0.028
      }
    }
  },
  "spill": {
    "name": "Spillvärme",
    "energies": {
      "electric": {
        "efficiency": 1.00,
        "quota": 0.042
```

```
      },
      "district_heating": {
        "efficiency": 1.00,
        "quota": 0.07
      }
    }
  }
},
"energies": {
  "fuel": {
    "name": "Drivmedel",
    "sectors": {
      "transport": {
        "efficiency": 0.20,
        "amount": 18
      }
    }
  },
  "electric": {
    "name": "El",
    "energies": {
      "heat_pump": {
        "efficiency": 0.90,
        "quota": 1.00
      },
      "heating_residences": {
        "efficiency": 0.90,
        "quota": 0.23
      }
    },
    "sectors": {
      "residences": {
        "efficiency": 0.90,
        "amount": 55
      },
      "industry": {
        "efficiency": 0.90,
        "amount": 53
      }
    }
  },
  "heat_pump": {
    "name": "Värmepump",
    "energies": {
```

```
        "district_heating": {
          "efficiency": 2.25,
          "quota": 0.22
        }
      }
    },
    "district_heating": {
      "name": "Fjärrvärme",
      "energies": {
        "heating_residences": {
          "efficiency": 0.81,
          "quota": 0.49
        },
        "heating_industry": {
          "efficiency": 0.81,
          "quota": 0.04
        }
      }
    },
    "heating_residences": {
      "name": "Värme, bostäder",
      "sectors": {
        "residences": {
          "efficiency": 1.00,
          "amount": 90
        }
      }
    },
    "heating_industry": {
      "name": "Värme, industri",
      "sectors": {
        "industry": {
          "efficiency": 1.00,
          "amount": 121
        }
      }
    }
  },
  "sectors": {
    "residences": {
      "name": "Bostäder"
    },
    "transport": {
      "name": "Transport"
```

```
    },
    "industry": {
      "name": "Industri"
    }
  }
}
```