```
> restart;
```

Supplementary Code for "Lubhani Mishra *et al* 2021 *J. Electrochem. Soc.* **168** 092502"

NOTE:

Correction: The definition of the Peclet number (Pe) has been modified to:

Pe = (Mw * i0 * L)/(rho * F * D)

Caveats:

This code takes the final form of the model after Landau transformation.

The model solves the PDEs for concentration and potential using numerical method of lines.

A 3 point forward/backward difference is used at the boundaries for first derivatives. Second order central difference is used in the spatial direction for second derivatives. A cell-centered finite difference/finite volume method may be more robust for general discretization of PDEs. In general, when convection (or first derivatives are involved) proper upwind schemes may be needed. N = 10 internal nodes points are used for demonstration. Higher number of node points may be needed for convergence.

This code is not optimized for speed/efficiency/memory usage.

The same model can be solved more efficiently using collocation in X (Code is available upon request).

Maple can't solve DAEs directly. So, the algebraic equations are differentiated and then solved using consistent initial condition found from Leastsquare minimization of residues. There are more robust approaches for solving battery models [M. T. Lawder, V. Ramadesigan, B. Suthar and V. R. Subramanian, “Extending explicit and linearly implicit ODE solvers for index-1 DAEs”, Computers and Chemical Engineering, 82, 283-292 (2015). USPTO 10769236].

Running the same code in C using SUNDIALS IDA will be 2 orders of magnitude faster.

Cycling codes (without documentation) is plotted here.

The equations below are the set of equations after transformation to non dimensional form.

Governing Equations

Equation for potential after Landau Transformation

```
> eq:=(diff(c(Z,tau),Z)*diff(phi(Z,tau),Z)+c(Z,tau)*diff(phi(Z,
  tau),Z$2))/(1-S(tau))^2=0;
```

$$eq := \frac{\left(\frac{\partial}{\partial Z} c(Z,\tau)\right)\left(\frac{\partial}{\partial Z} \phi(Z,\tau)\right) + c(Z,\tau)\left(\frac{\partial^2}{\partial Z^2} \phi(Z,\tau)\right)}{(1-S(\tau))^2} = 0 \quad (1)$$

Equation for concentration after Landau transformation

```
> ceq:=diff(c(Z,tau),tau)=-Pe/tD*CC[0](tau)^.5*(Va(tau)-UU[0](tau))
  *(1-Z)/(1-S(tau))*diff(c(Z,tau),Z)+diff(c(Z,tau),Z$2)/(tD*(1-S
  (tau))^2);
```

$$ceq := \frac{\partial}{\partial \tau} c(Z,\tau) = -\frac{Pe\sqrt{CC_0(\tau)}\,\left(Va(\tau) - UU_0(\tau)\right)(1-Z)\left(\frac{\partial}{\partial Z} c(Z,\tau)\right)}{tD\,(1-S(\tau))} + \frac{\frac{\partial^2}{\partial Z^2} c(Z,\tau)}{tD\,(1-S(\tau))^2} \quad (2)$$

Boundary Conditions

Boundary condition for potential at X=0

```
> bc1:=diff(phi(Z,tau),Z)/(1-S(tau))=-Da/c(Z,tau)^.5*(Va(tau)-phi
  (Z,tau));
```

$$bc1 := \frac{\frac{\partial}{\partial Z}\,\phi(Z,\tau)}{1-S(\tau)} = -\frac{Da\,(Va(\tau)-\phi(Z,\tau))}{\sqrt{c(Z,\tau)}} \tag{3}$$

Boundary conditions for concentration at X=0

```
> cbc1:=diff(c(Z,tau),Z)/(1-S(tau))=-Da*c(Z,tau)^.5*(Va(tau)-phi(Z,
  tau))+cv*Pe*c(Z,tau)^1.5*(Va(tau)-phi(Z,tau));
```

$$cbc1 := \frac{\frac{\partial}{\partial Z}\,c(Z,\tau)}{1-S(\tau)} = -Da\sqrt{c(Z,\tau)}\,(Va(\tau)-\phi(Z,\tau)) + cv\,Pe\,c(Z,\tau)^{1.5}\,(Va(\tau)-\phi(Z,\tau)) \tag{4}$$

Boundary condition for potential at X=1

```
> bc2:=c(Z,tau)*diff(phi(Z,tau),Z)=-delta*(1-S(tau));
```

$$bc2 := c(Z,\tau)\left(\frac{\partial}{\partial Z}\,\phi(Z,\tau)\right) = -\delta\,(1-S(\tau)) \tag{5}$$

Boundary condition for concentration at X=1

```
> cbc2:=diff(c(Z,tau),Z)=-delta*(1-S(tau));
```

$$cbc2 := \frac{\partial}{\partial Z}\,c(Z,\tau) = -\delta\,(1-S(\tau)) \tag{6}$$

List of parameters

MV = Mw/rho

```
> pars0:=[iapp=10, i0=20,i0cathode=20, D1=1e-11, F= 96487, R=8.314,
  T=298, Lx=50e-6, c0=1000, MV= 1.2998e-5];
```

$$pars0 := [iapp = 10,\ i0 = 20,\ i0cathode = 20,\ D1 = 1.\times 10^{-11},\ F = 96487,\ R = 8.314,\ T = 298,\ Lx = 0.000050,\ c0 = 1000,\ MV = 0.000012998] \tag{7}$$

Dimensionless groups

```
> pars:=subs(pars0,[
  delta=iapp*Lx*Crate/2/F/D1/c0,
  tD=abs(Crate)*Lx^2/(3600*D1),
  Da=i0*Lx/(2.*F*D1*c0),
  Pe=MV*i0*Lx/(D1*F),
  phi0=R*T/F]);
```

$$pars := [\delta = 0.2591022625\ Crate,\ tD = 0.06944444444\ |Crate|,\ Da = 0.5182045250,\ Pe = 0.01347124483,\ \phi 0 = 0.02567778043] \tag{8}$$

Discretizing the equations using finite difference

Input number of node points

```
> N1:=10;
```

$$N1 := 10 \tag{9}$$

Node spacing

```
> h:=1.0/(N1+1);
```

(10)

$$h := 0.09090909091 \quad (10)$$

The variables of c and phi are renamed to be u1 and u2, where u1 = c, u2 = phi

```
> #u1 = C, u2 = phi #
```

eq

```
> for i from 1 to N1 do:
  eq11[i]:=(u1[i+1](tau)-u1[i-1](tau))/(2*h)*(u2[i+1](tau)-u2[i-1]
  (tau))/(2*h)/(-1+S(tau))^2+u1[i](tau)*(u2[i-1](tau)-2*u2[i](tau)+
  u2[i+1](tau))/h^2/(-1+S(tau))^2=0;
  od:
```

Discretized boundary conditions

bc1

```
> eq11[0]:=(-3*u2[0](tau)+4*u2[1](tau)-u2[2](tau))/(2*h)=-1*Da*(Va
  (tau)-u2[0](tau))*(1-S(tau))/u1[0](tau)^.5;
```

$$eq11_0 := -16.50000000\,u2_0(\tau) + 22.00000000\,u2_1(\tau) - 5.500000000\,u2_2(\tau) = -\frac{Da\left(Va(\tau) - u2_0(\tau)\right)\left(1 - S(\tau)\right)}{\sqrt{u1_0(\tau)}} \quad (11)$$

bc2

```
> eq11[N1+1]:=(-3*u2[N1+1](tau)+4*u2[N1](tau)-u2[N1-1](tau))/(2*h)=
  -1*delta*(1-S(tau))/u1[N1+1](tau);
```

$$eq11_{11} := -16.50000000\,u2_{11}(\tau) + 22.00000000\,u2_{10}(\tau) - 5.500000000\,u2_9(\tau) = -\frac{\delta\left(1 - S(\tau)\right)}{u1_{11}(\tau)} \quad (12)$$

ceq

```
> ceq;
```

$$\frac{\partial}{\partial\tau}\,c(Z,\tau) = -\frac{Pe\sqrt{CC_0(\tau)}\left(Va(\tau) - UU_0(\tau)\right)(1 - Z)\left(\frac{\partial}{\partial Z}\,c(Z,\tau)\right)}{tD\left(1 - S(\tau)\right)} + \frac{\frac{\partial^2}{\partial Z^2}\,c(Z,\tau)}{tD\left(1 - S(\tau)\right)^2} \quad (13)$$

```
> for i from 1 to N1 do:
  eq21[i]:=diff(u1[i](tau),tau)=Pe/tD*u1[0](tau)^.5*(u2[0](tau)-Va
  (tau))*(1-i*h)/(2*h)*(u1[i+1](tau)-u1[i-1](tau))/(1-S(tau)) + (u1
  [i-1](tau)-2*u1[i](tau)+u1[i+1](tau))/h^2/(tD*(1-S(tau))^2);od:
```

cbc1

```
> eq21[0]:=(-3*u1[0](tau)+4*u1[1](tau)-u1[2](tau))/(2*h)=(-Da*u1[0]
  (tau)^.5*(Va(tau)-u2[0](tau))+cv*Pe*u1[0](tau)^1.5*(Va(tau)-u2[0]
  (tau)))*(1-S(tau));
```

$$eq21_0 := -16.50000000\,u1_0(\tau) + 22.00000000\,u1_1(\tau) - 5.500000000\,u1_2(\tau) = \left(\quad (14)$$

$$-Da\sqrt{uI_0(\tau)}\left(Va(\tau)-u2_0(\tau)\right)+cv\,Pe\,uI_0(\tau)^{1.5}\left(Va(\tau)-u2_0(\tau)\right)\Big)\left(1-S(\tau)\right)$$

cbc2

```
> eq21[N1+1]:=(-3*u1[N1+1](tau)+4*u1[N1](tau)-u1[N1-1](tau))/(2*h)=
  -delta*(1-S(tau));
```

$$eq21_{11} := -16.50000000\,uI_{11}(\tau)+22.00000000\,uI_{10}(\tau)-5.500000000\,uI_9(\tau)=-\delta\left(1-S(\tau)\right) \quad (15)$$

Reference electrode potential

```
> Va(tau):=0;
```

$$Va(\tau) := 0 \quad (16)$$

Moving boundary velocity

```
> eqz1:=diff(S(tau),tau)=-Pe/tD*(Va(tau)-u2[0](tau))*u1[0](tau)
  ^0.5;
```

$$eqz1 := \frac{d}{d\tau}S(\tau)=\frac{Pe\,u2_0(\tau)\sqrt{uI_0(\tau)}}{tD} \quad (17)$$

Switch to enable/disable the convective term in the boundary equation cbc1
cv = 1 : enable / 0 : disable

```
> cv:=1;
```

$$cv := 1 \quad (18)$$

Separating the Ordinary Differential Equations (ODEs) and the Algebraic Equations (AEs)

```
> eqodes:=subs(pars,pars0,[seq(eq21[i],i=1..N1),eqz1]):
> eqaes :=subs(pars,pars0,[seq(eq11[i],i=0..N1+1),eq21[0],eq21
  [N1+1]]):
> eqaes:=map(rhs-lhs,eqaes):
> eqaes2:=subs(eqodes,diff(eqaes,tau)):
```

List of variables

```
> aevars:=[seq(u2[i](tau),i=0..N1+1),u1[0](tau),u1[N1+1](tau)]:
> odevars:=[seq(u1[i](tau),i=1..N1),S(tau)]:
> aesubs:=[seq(aevars[i]=ZZ[i],i=1..nops(aevars))]:
```

```
> single_phase:=proc(cycle,phase):
  global sol,dsol;
  local i,rate,icodes,eqic,solic,icaes,ics;
  if   phase = c then rate:= 1;
  elif phase = d then rate:=-1;
  fi:
```

```
  if cycle = 1 and phase = c then icodes:=[seq(u1[i](tau)=1,i=1..
  N1),S(tau)=.02]:
  elif phase = c then icodes:=[seq(odevars[i]=subs(sol[cycle-1,d],
  odevars[i]),i=1..N1+1)]:
  else                icodes:=[seq(odevars[i]=subs(sol[cycle  ,c],
  odevars[i]),i=1..N1+1)]:
  fi:
  eqic:=subs(icodes,aesubs,Crate=rate,eqaes):
  solic:=Optimization:-LSSolve(eqic):
  icaes:=subs(solic[2],aesubs):
  ics:=op(subs(tau=0,icodes)),op(subs(tau=0,icaes)):
  dsol:=dsolve({op(subs(Crate=rate,eqodes)),op(subs(Crate=rate,
  eqaes2)),ics},type=numeric,implicit=true,maxfun=0,stiff=true,
  compile=true):
  sol[cycle,phase]:=dsol(1):
  end proc:
```

```
> makeplot:=proc(cycle,phase)
  global p1,p2,p3,p4,p5;
  local offset;
  if phase = d then offset:=1; else offset:=0; fi:
  p1[cycle,phase]:=plots:-odeplot(dsol,[(2*(cycle-1)+offset+tau)*
  3600,u1[   0](tau)*1000],0..1,axes=boxed,legend=  C_at_MB ,
  thickness=3,color=red ):
  p2[cycle,phase]:=plots:-odeplot(dsol,[(2*(cycle-1)+offset+tau)*
  3600,u1[N1+1](tau)*1000],0..1,axes=boxed,legend=  C_at_top,
  thickness=3,color=blue):
  p3[cycle,phase]:=plots:-odeplot(dsol,[(2*(cycle-1)+offset+tau)*
  3600,u2[   0](tau)     ],0..1,axes=boxed,legend=Phi_at_MB ,
  thickness=3,color=red ):
  p4[cycle,phase]:=plots:-odeplot(dsol,[(2*(cycle-1)+offset+tau)*
  3600,u2[N1+1](tau)     ],0..1,axes=boxed,legend=Phi_at_top,
  thickness=3,color=blue):
  p5[cycle,phase]:=plots:-odeplot(dsol,[(2*(cycle-1)+offset+tau)*
  3600,       S(tau)*50  ],0..1,axes=boxed,thickness=3,color=red):
  print(Cycle=cycle,Phase=phase,ended);
  end proc:
```

```
> runcycle:=proc(cycle)
  global pc,pphi,pmb;
  local i;
  for i from 1 to cycle do
```

```
single_phase(i,c);
    makeplot(i,c);
single_phase(i,d);
    makeplot(i,d);
od:

pc  :=plots:-display({seq(op([p1[i,c],p1[i,d],p2[i,c],p2[i,d]]),
i=1..testcycle)},title="Li Concentration during charging phase",
                    labels=["time [seconds]","Li concentration
[mol/m^3]"],labeldirections=[horizontal, vertical]):
pphi:=plots:-display({seq(op([p3[i,c],p3[i,d],p4[i,c],p4[i,d]]),
i=1..testcycle)},title="Electrolyte potential during charging
phase",
                    labels=["time [seconds]","Potential [V]"],
labeldirections=[horizontal, vertical]):
pmb :=plots:-display({seq(op([p5[i,c],p5[i,d]]),i=1..testcycle)},
title="Li Electrode Thickness during charging phase",
                    labels=["time [seconds]","Thickness [1e-6 m]
"],labeldirections=[horizontal, vertical]):
print("simulation complete"):
end proc:
```

```
> testcycle:=10;
```

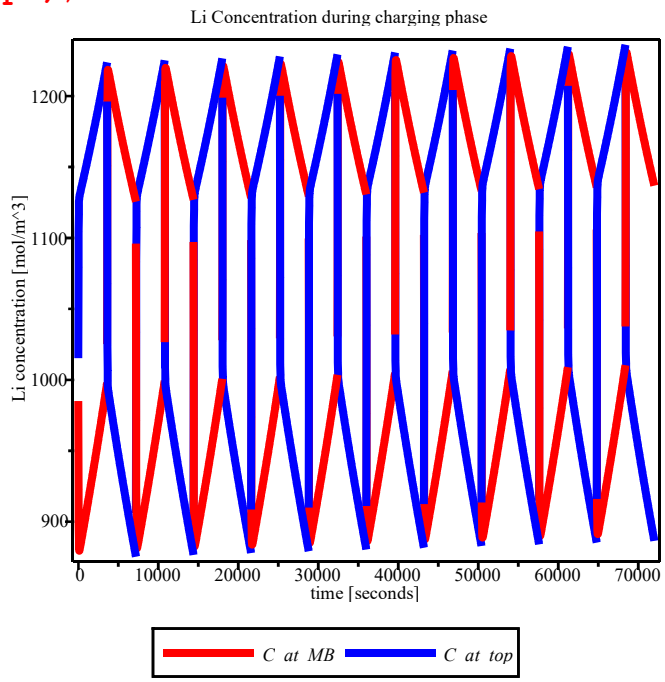
$$testcycle := 10 \tag{19}$$

```
> runcycle(testcycle);
```

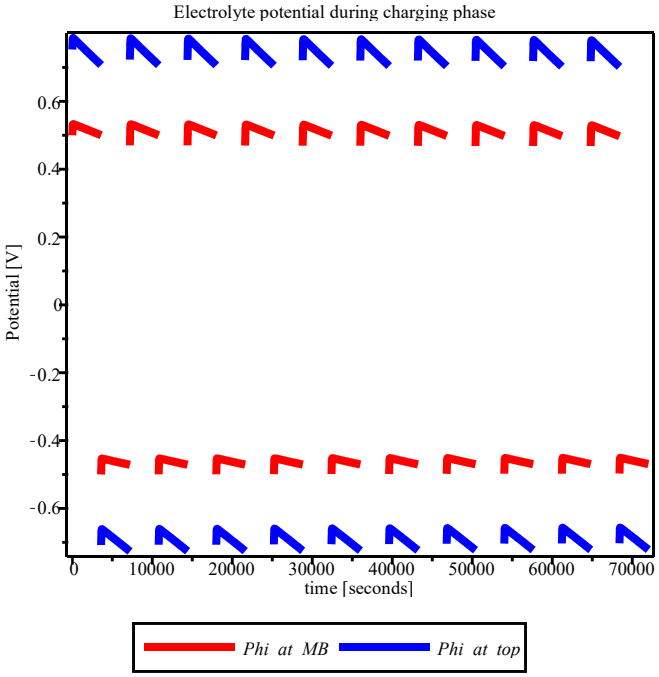
$Cycle = 1, Phase = c, ended$
$Cycle = 1, Phase = d, ended$
$Cycle = 2, Phase = c, ended$
$Cycle = 2, Phase = d, ended$
$Cycle = 3, Phase = c, ended$
$Cycle = 3, Phase = d, ended$
$Cycle = 4, Phase = c, ended$
$Cycle = 4, Phase = d, ended$
$Cycle = 5, Phase = c, ended$
$Cycle = 5, Phase = d, ended$
$Cycle = 6, Phase = c, ended$
$Cycle = 6, Phase = d, ended$
$Cycle = 7, Phase = c, ended$
$Cycle = 7, Phase = d, ended$
$Cycle = 8, Phase = c, ended$
$Cycle = 8, Phase = d, ended$
$Cycle = 9, Phase = c, ended$

Cycle = 9, *Phase* = *d*, *ended*
Cycle = 10, *Phase* = *c*, *ended*
Cycle = 10, *Phase* = *d*, *ended*
"simulation complete" **(20)**

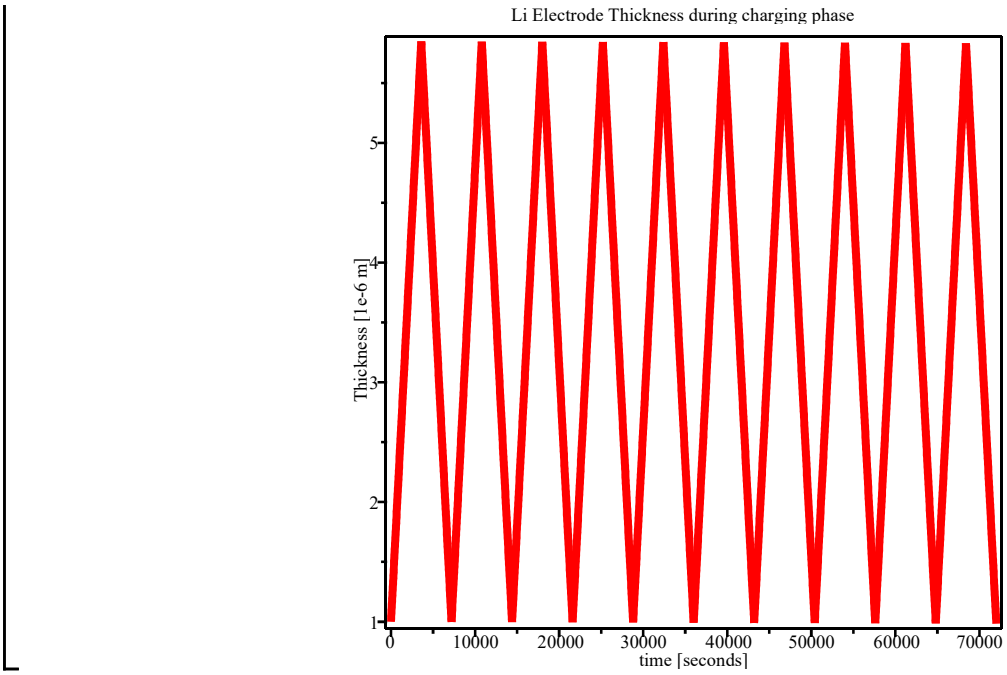
```
> plots:-display(pc);
```




```
> plots:-display(pphi);
```




```
> plots:-display(pmb);
```


Li Electrode Thickness during charging phase
Thickness [1e-6 m]
1
2
3
4
5
0
10000
20000
30000
40000
50000
60000
70000
time [seconds]