AugerBot Calculations

```
Quit
```

Trial 1: 9/19 - 9/26

Modified Francisco Calculations : 10/18 (Do not use!)

Plotting Fx to find U which Balances Forces: 11/1

Plotting Fx to find U which Balances Forces 2 : 11/9

Inside Equation for Thrust: 11/16

```
In[35]:= Quit;
```

Testing Model

Parameters

For Helix

```
In[1]:= (*Current param: R = 1.8cm, n = 3.5*)
R = 0.018; (*Screw radius, m*)
n = 3.5; (*Number of helix turns*)
```

For Material

In[3]:=
```
(*S5 and S6:
   az(β,γ) = az(0,sgn(zdot)*π/2)*|Cos[β]|, Approx Fz on horiz projection
   ax(β,γ) = ax(π/2,0)*|Sin[β]|, Approx Fx on vert projection
*)

(*NOT SURE IF ZDOT IS + OR -, IF - THEN ALL Z VALUES UNDER AROUND 0*)
zlpPoppy = 0.35;
zcpPoppy = 0.55;
zlpGlass = 0.3;
zcpGlass = 0.4;
zcpGLASS = 0.3;(*az(0,sgn(zdot)*π/2), N/cm^3*)
xlpPoppy = 0.0625;
xcpPoppy = 0.2/3;
xlpGlass = 0.075;
xcpGlass = 0.1;
xcpGLASS = 0.0625;(*ax(π/2,0), N/cm^3*)

β = (Pi/2) - ϕ; (*ϕ is symbolic, radians*)
αz = zlpPoppy * Abs[Cos[β]] * (100^3); (*Vertical stress per unit depth, N/m^3*)
αx = xlpPoppy * Abs[Sin[β]] * (100^3); (*Horizontal stress per unit depth, N/m^3*)

d = 0.05;(*Depth robot buried, m*)

(*Friction coefficients, expressed in terms of ϕ*)
Cn = αz * d; (*N/m^2*)
Ct = αx * d;
```

For Motor

In[11]:=
```
w = 2 * 1000 * (2 * Pi) / 3584;(*Angular velocity with 12V source, rad/s*)
```

- (2 ticks/ms)*(1000 ms/s)*(2*Pi rad/rev)*(1 rev/3584 ticks)

Horizontal Thrust Inner Equation

In[12]:=
```
FxIn[u_] :=
  (0.5 * (Cn - Ct) * Sin[ϕ] * Cos[ϕ] * (Sqrt[1 + u^2] + u^2 * Log[u / (1 + Sqrt[1 + u^2])]) -
    u * (Ct * Sin[ϕ]^2 + Cn * Cos[ϕ]^2) * (Sqrt[1 + u^2] - u));
```

ϕ input must be in radians

Calculating U/Rw when Fx = 0 for Many ϕ Cases

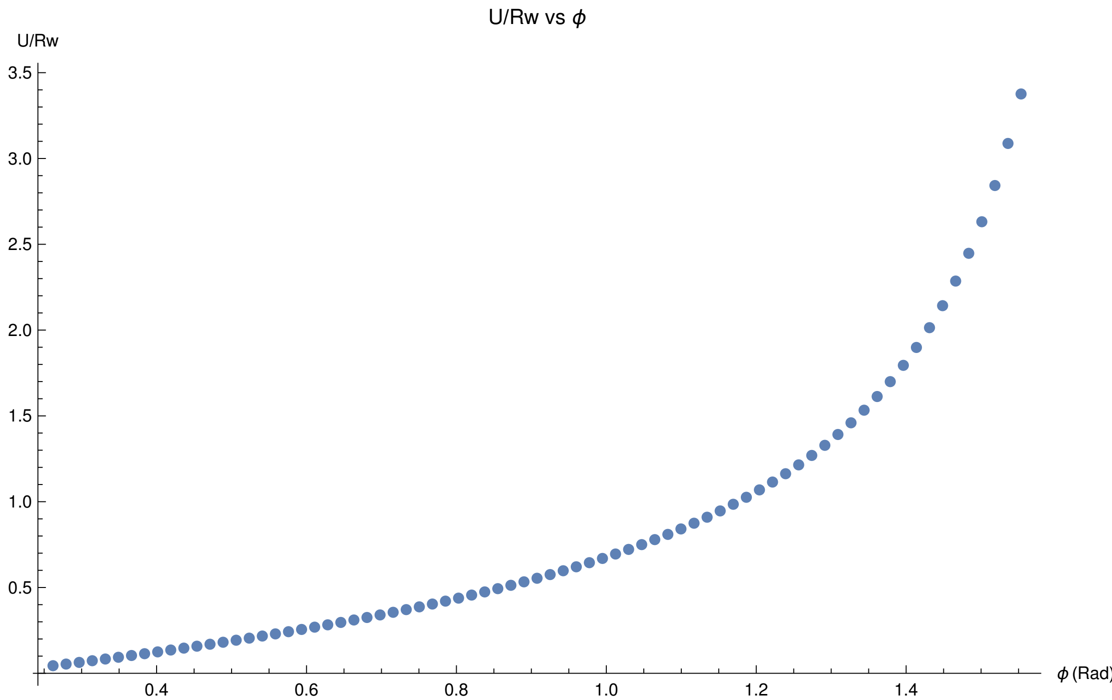
```
In[13]:= ϕ = 15 * Pi / 180 // N; (*Local inclination, radians*)
ϕstore = {};
ustore = {};
Fstore = {};
FMaxstore = {}; (*FxIn when u = 0.001*)
```

```
In[18]:= (*Finding U/Rw Intercepts*)
While[ϕ < 90 * Pi / 180,
 (*Print statements*)
 (*Print["Let ϕ = ",ϕ*180/Pi, " deg"];
 Print["(Cn-Ct)*Sin[ϕ]*Cos[ϕ] = ",(Cn-Ct)*Sin[ϕ]*Cos[ϕ]];
 Print["Ct*Sin[ϕ]^2+Cn*Cos[ϕ]^2 = ",Ct*Sin[ϕ]^2+Cn*Cos[ϕ]^2];*)
 (*Print@Plot[{FxIn[u],0},{u,0,0.1}, PlotLabel→"Inner Fx vs. Helix Velocity U/Rw",
    AxesLabel→{"U/Rw (m/s)","Fx (N/m^2)"}, PlotRange→All];*)

 (*Finding U/Rw intercept: Newton-Raphson Method*)
 guess = 0.01; (*Reset initial guess*)
 grad = D[FxIn[u], u];
 While[FxIn[guess] > 10^-6, (*Keep iterating until FxIn ≃ 0*)
  gradEval = grad /. u → guess; (*Find FxIn'(guess)*)
  guess = guess - FxIn[guess] / gradEval (*u_{i+1} = u_i - FxIn(u_i)/FxIn'(u_i)*)
 ];
 uint = guess; (*U/Rw intercept found*)

 (*Storing data in arrays*)
 ϕstore = Join[ϕstore, {ϕ}]; (*Storing ϕ in Radians*)
 FMaxstore = Join[FMaxstore, {FxIn[10^-3]}]; (*FxIn(0.001)*)
 ustore = Join[ustore, {uint}]; (*U/Rw found when FxIn < 10^-6*)
 Fstore = Join[Fstore, {FxIn[uint]}]; (*FxIn val at u-intercept*)

 ϕ = ϕ + (1 * Pi / 180) (*Increment by 1 deg*)
]
```

Analysis

In[19]:=
```
ϕcol = {ϕstore}ᵀ;
ucol = {ustore}ᵀ;
Fcol = {Fstore}ᵀ;
Fmaxcol = {FMaxstore}ᵀ;

(*Helix Translation Speed vs. Local Incline Angle ϕ*)
dataPlot = Join[ϕcol, ucol, 2];
ListPlot[dataPlot, PlotLabel → "U/Rw vs ϕ",
 AxesLabel → {"ϕ (Rad)", "U/Rw"}, ImageSize → Large]
```

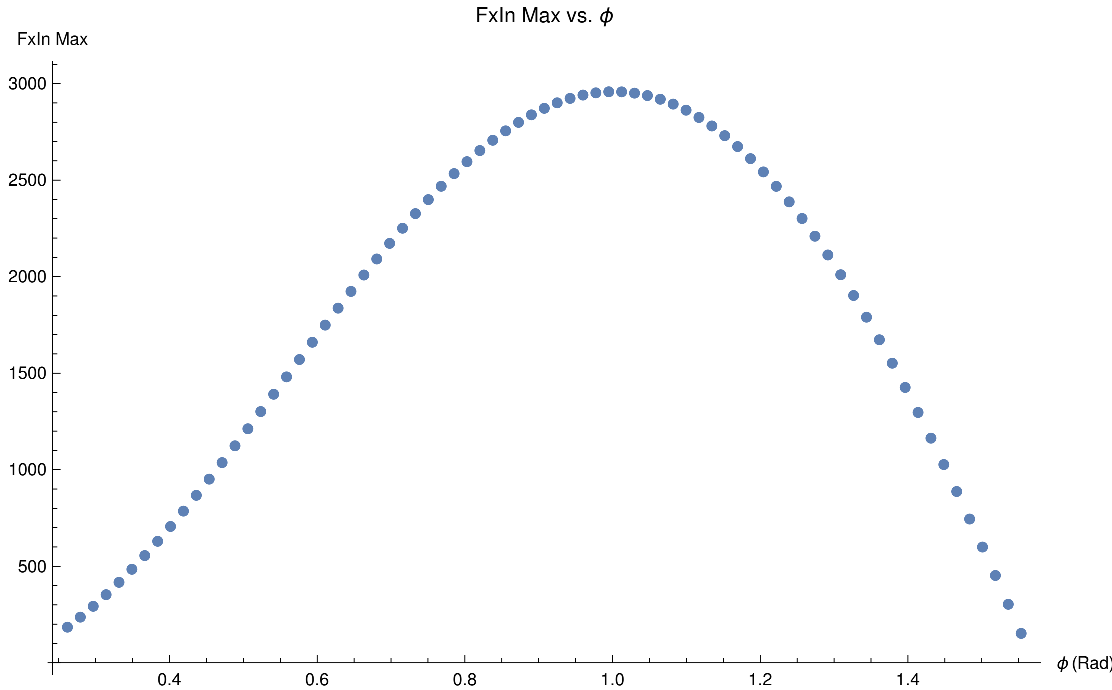
Out[21]=



Should have a maximum.

In[22]:=
```
(*FxIn Max vs. Local Incline Angle ϕ*)
dataMax = Join[ϕcol, Fmaxcol, 2];
ListPlot[dataMax, PlotLabel → "FxIn Max vs. ϕ",
 AxesLabel → {"ϕ (Rad)", "FxIn Max"}, ImageSize → Large, PlotRange → Full]
```

Out[23]=



In[24]:=
```
(*Checking Data Values: ϕ is left col, U/Rw is right col*)
dataTable = Join[ϕcol, ucol, Fcol, 2];
Grid[Join[{{"ϕ (Deg)", "U/Rw", "FxIn[U/Rw]"}}, dataTable, 1]]
```

ϕ (Deg)	U/Rw	FxIn[U/Rw]
0.261799	0.0442894	3.55897×10^{-9}
0.279253	0.0538053	2.30265×10^{-8}
0.296706	0.0634723	1.036×10^{-7}
0.314159	0.0732932	3.64453×10^{-7}
0.331613	0.0832716	5.68434×10^{-14}
0.349066	0.0934114	5.68434×10^{-14}
0.366519	0.103717	1.13687×10^{-13}
0.383972	0.114193	0.
0.401426	0.124845	1.13687×10^{-13}
0.418879	0.135678	4.54747×10^{-13}
0.436332	0.146699	1.36424×10^{-12}
0.453786	0.157913	3.86535×10^{-12}
0.471239	0.169329	1.01181×10^{-11}
0.488692	0.180953	2.54659×10^{-11}
0.506145	0.192792	6.09361×10^{-11}
0.523599	0.204857	1.38243×10^{-10}
0.541052	0.217155	2.98769×10^{-10}
0.558505	0.229697	6.2164×10^{-10}
0.575959	0.242491	1.24646×10^{-9}

Out[25]=

0.593412	0.255551	2.42062×10^{-9}
0.610865	0.268885	4.56021×10^{-9}
0.628319	0.282509	8.36349×10^{-9}
0.645772	0.296433	1.49571×10^{-8}
0.663225	0.310673	2.61371×10^{-8}
0.680678	0.325243	4.47076×10^{-8}
0.698132	0.34016	7.49646×10^{-8}
0.715585	0.355439	1.2339×10^{-7}
0.733038	0.3711	1.99614×10^{-7}
0.750492	0.387163	3.17728×10^{-7}
0.767945	0.403647	4.98091×10^{-7}
0.785398	0.420575	7.69727×10^{-7}
0.802851	0.437972	-4.54747×10^{-13}
0.820305	0.455864	-4.54747×10^{-13}
0.837758	0.474277	-4.54747×10^{-13}
0.855211	0.493243	0.
0.872665	0.512794	4.54747×10^{-13}
0.890118	0.532964	0.
0.907571	0.553792	0.
0.925025	0.575319	4.54747×10^{-13}
0.942478	0.59759	0.
0.959931	0.620653	2.27374×10^{-13}
0.977384	0.644562	4.54747×10^{-13}
0.994838	0.669374	6.82121×10^{-13}
1.01229	0.695153	2.04636×10^{-12}
1.02974	0.721969	3.63798×10^{-12}
1.0472	0.749898	5.68434×10^{-12}
1.06465	0.779024	1.11413×10^{-11}
1.0821	0.809441	2.02363×10^{-11}
1.09956	0.841251	3.66072×10^{-11}
1.11701	0.874571	6.61657×10^{-11}
1.13446	0.909526	1.14824×10^{-10}
1.15192	0.946259	1.99179×10^{-10}
1.16937	0.984931	3.39924×10^{-10}
1.18682	1.02572	5.74346×10^{-10}
1.20428	1.06883	9.63837×10^{-10}
1.22173	1.11449	1.59594×10^{-9}
1.23918	1.16297	2.61389×10^{-9}
1.25664	1.21455	4.23711×10^{-9}
1.27409	1.26958	6.79756×10^{-9}
1.29154	1.32846	1.0794×10^{-8}
1.309	1.39165	1.69572×10^{-8}
1.32645	1.45967	2.63642×10^{-8}
1.3439	1.53316	4.05579×10^{-8}
1.36136	1.61285	6.17229×10^{-8}
1.37881	1.69962	9.28998×10^{-8}
1.39626	1.79452	1.38224×10^{-7}
1.41372	1.89883	2.03169×10^{-7}

1.43117	2.0141	2.94674×10^{-7}
1.44862	2.14226	4.20971×10^{-7}
1.46608	2.28569	5.90566×10^{-7}
1.48353	2.44742	8.09265×10^{-7}
1.50098	2.63133	-2.84217×10^{-13}
1.51844	2.8425	2.84217×10^{-13}
1.53589	3.08767	-5.18696×10^{-13}
1.55334	3.376	-4.79616×10^{-13}

In[26]:=
```
ϕ = ϕcol[[5, 1]]
u0 = ucol[[5, 1]]
FxIn[u0]
```

Out[26]= 0.331613

Out[27]= 0.0832716

Out[28]= 5.68434×10^{-14}